The Dissertation Committee for Samuel Clay Estes certifies that this is the approved version of the following dissertation:

# Uncertainty Quantification in Reservoirs with Faults Using a Sequential Approach

**Committee:**

Clint Dawson, Supervisor

Tan Bui-Thanh

Omar Ghattas

Troy Butler

George Biros

# Uncertainty Quantification in Reservoirs with Faults Using a Sequential Approach

by

**Samuel Clay Estes**

**Dissertation**

Presented to the Faculty of the Graduate School
of the University of Texas at Austin
in Partial Fulfillment
of the Requirements
for the Degree of

**Doctor of Philosophy**

The University of Texas at Austin
December 2019

Dedicated to my parents, Tommy and Rebekah Estes. I could never have done this without your love and support.

## Acknowledgments

First of all, I would like to thank my parents and the rest of my family for supporting me throughout this process. I would also like to thank my advisor Dr. Clint Dawson for helping me to become a more effective independent researcher. There are too many friends who helped me get to this point to mention them all by name. I would like to give special mention to Teresa, Sid, Tim, Gopal, and Prakash for their invaluable advice and help when writing this dissertation.

We gratefully acknowledge support from ExxonMobil Upstream Research Company, particularly Thomas Halsey and Pengbo Lu.

# Uncertainty Quantification in Reservoirs with Faults Using a Sequential Approach

by

Samuel Clay Estes, Ph.D.
The University of Texas at Austin, 2019
SUPERVISOR: Clint Dawson

Reservoir simulation is critically important for optimally managing petroleum reservoirs. Often, many of the parameters of the model are unknown and cannot be measured directly. These parameters must then be inferred from production data at the wells. This is an inverse problem which can be formulated within a Bayesian framework to integrate prior knowledge with observational data. Markov Chain Monte Carlo (MCMC) methods are commonly used to solve Bayesian inverse problems by generating a set of samples which can be used to characterize the posterior distribution.

In this work, we present a novel MCMC algorithm which uses a sequential transition kernel designed to exploit the redundancy which is often present in time series data from reservoirs. This method can be used to efficiently generate samples from the Bayesian posterior for time-dependent models. While this method is general and could be useful for many different models, we focus here on reservoir models with faults. A fault is a heterogeneity characterized by a sharp change in the permeability of the reservoir over a region with very small width relative to its length and the overall size of the reservoir domain [1]. It is often convenient to model faults as lower dimensional surfaces which act as barriers to the flow. The transmissibility multiplier is a parameter which characterizes the extent to which fluid

can flow across a fault. We consider a Bayesian inverse problem in which we wish to infer fault transmissibilities from measurements of pressure at wells using a two-phase flow model. We demonstrate how the sequential MCMC algorithm presented here can be more efficient than a standard Metropolis-Hastings MCMC approach for this inverse problem. We use integrated autocorrelation times along with mean-squared jump distances to determine the performance of each method for the inverse problem.

# Contents

# Chapter 1

# Introduction

## 1.1 Background and Motivation

Accurate simulation of flow in porous media is critically important in many applications such as groundwater contaminant transport, carbon sequestration, and petroleum production. We focus here on applications to petroleum reservoir simulation. Reservoir simulation can be used to optimize utilization of current oil fields and reduce exploration in vulnerable areas such as the Arctic regions [2]. However, obtaining accurate reservoir models can be extremely challenging due to the difficulty in directly measuring model parameters such as the permeability field or transmissibility of faults directly. This challenge is addressed by the field of history matching. History matching is a type of inverse problem in which observational data collected from a reservoir is used to estimate model parameters which cannot be directly observed [3]. Unfortunately, obtaining history matched models can be extremely expensive due to the large number of model parameters, nonlinearity of the model, and the high computational cost of running even a single reservoir simulation. For this reason, developing algorithms which take advantage of the unique structure of the history matching inverse problem is extremely important.

It is often convenient to pose the history matching problem as a Bayesian inverse problem. The solution to this problem is then a probability distribution known as the posterior,

which combines prior knowledge about model parameters with information from (noisy) observational data. In practice, obtaining useful information from this posterior probability can be difficult and it is common to employ Monte Carlo methods for estimating moments of the posterior. In particular, Markov chain Monte Carlo (MCMC) methods are often used to generate samples which are distributed (approximately) according to the posterior as they tend to suffer from the Curse of Dimensionality less than some other methods. However, MCMC methods can be prohibitively expensive for many history matching applications due to the difficulties outlined above.

In this work, we present an MCMC algorithm which is designed to exploit the structure inherent in many history matching problems and apply it to a two-phase reservoir model. The algorithm here relies on a novel sequential transition kernel which may reduce the number of model solves required by MCMC. The key assumption behind the approach is that the time series of production data in reservoirs is often somewhat redundant. To be more precise, consider two vectors of model parameters $x_1$ and $x_2$ with corresponding time series model output $\{d_1^t\}_{t=1}^T$ and $\{d_2^t\}_{t=1}^T$. The assumption is that for many parameters $x_1$ and $x_2$, if $d_1^1$ more closely matches the observed reservoir data than $d_2^1$ at time $t = 1$, then the rest of the time-series $\{d_1^t\}_{t=2}^T$ will as well. Although this assumption may not hold exactly for every pair of parameters and is complicated by the presence of noise in the data, we show that it can still be leveraged to implement an algorithm which yields increased efficiency over a simple yet commonly used implementation of MCMC in practical circumstances.

## 1.2 Outline

The outline of this work is as follows. In Chapter 2 we discuss the details of the reservoir model used in this work and how it is implemented numerically. In Chapter 3 we review history matching in reservoirs and describe the Bayesian formulation for inverse problems along

with the popular Metropolis-Hastings (MH)-MCMC algorithm. In Chapter 4 we present the sequential transition kernel for MCMC used in this work. We also describe how we compare the efficiency of the sequential approach to a standard MH-MCMC approach and apply both methods to a simple example to illustrate an important situation in which it is useful. In Chapter 5 we apply the sequential MCMC algorithm to an inverse problem defined by the two-phase flow model with faults. Performance is compared to a standard MH-MCMC algorithm. Finally, in Chapter 6 we discuss the results of this thesis and consider future work.

# Chapter 2

# Reservoir Models with Faults

Reservoir simulation involves solving the equations governing the flow of fluid in a porous medium in order to predict future production of hydrocarbons. In this work, we model the flow of fluids in a reservoir, namely water and oil, using a two-phase compressible flow model. The reservoir is assumed to have faults which are modelled as lower dimensional objects that introduce a discontinuous jump in the pressure field and act as barriers to the flow. The transmissibility of the fault is a parameter which relates the jump in pressure to the flow velocity normal to the fault. The equations are discretized using a finite element formulation in space and an implicit Euler scheme in time. In order to solve the resulting discretized system we use FEniCS, an open source library for solving finite element systems. In this section, we first summarize the equations of two-phase flow in a porous medium. For a more detailed description of porous media flow, see [2, 4]. Second, the representation of faults as lower dimensional objects is disussed and fault transmissibilities are introduced. Modelling fractures and faults as lower dimensional interfaces in reservoir simulation has been discussed in [5–7]. Finally, the numerical implementation of the forward model using the FEniCS library is described. More information about the FEniCS library can be found in [8].

## 2.1 Two-Phase Flow

Two-phase immiscible flow of Newtonian fluids in a porous medium under isothermal conditions is governed by an equation for conservation of mass and Darcy's Law for each phase. The mass conservation equation for each phase is given by:

$$\frac{\partial \left(\phi \rho_\alpha S_\alpha\right)}{\partial t} = -\nabla \cdot \left(\rho_\alpha \, \mathbf{u}_\alpha\right) + \rho_\alpha \, q_\alpha, \qquad \alpha = w, n, \tag{2.1}$$

where $\alpha = w, n$ for the wetting and non-wetting phases respectively, $\phi$ is the porosity of the porous medium, $\rho_\alpha$ is the density of each phase, $S_\alpha$ is saturation of each phase, $\mathbf{u}_\alpha$ is the Darcy velocity of each phase, and $q_\alpha$ is the external source/sink term for each phase. Here we assume that the wetting phase is water while the non-wetting phase is oil. The momentum conservation equation for each phase is given by Darcy's law:

$$\mathbf{u}_\alpha = -\frac{1}{\mu_\alpha} \mathbf{k}_\alpha \left(\nabla p_\alpha - \rho_\alpha \mathbf{g} \nabla \mathbf{z}\right), \qquad \alpha = w, n, \tag{2.2}$$

where $\mathbf{k}_\alpha$ is the effective permeability for each phase, $p_\alpha$ is the pressure for each phase, and $\mu_\alpha$ is the viscosity for each phase. The difference in the phase pressures is given by the capillary pressure $p_c = p_n - p_w$. Empirically, the capillary pressure is some known function of the saturations. Throughout, we assume that the capillary pressure is negligible and define a global pressure $p = p_n = p_o$.

Additional equations are required for the phase saturations, effective permeabilities, and rock and fluid compressibilities. Together, both phases fill the entire pore volume so that

$$S_n + S_w = 1. \tag{2.3}$$

The effective permeability for each phase $\mathbf{k}_\alpha$ is related to the absolute permeability of the

porous medium $\mathbf{k}$ by the relative permeabilities $k_{r\alpha}$

$$\mathbf{k}_\alpha = k_{r\alpha}\mathbf{k}, \tag{2.4}$$

where $k_{r\alpha}$ is assumed to be some known function of the saturation. Finally, the fluid densities $\rho_\alpha$ and the porosity of the medium $\phi$ may depend on the pressure. To model this dependence, the rock and fluid compressibilities are introduced. These are given by

$$c_r = \frac{1}{\phi}\frac{d\phi}{dp} \text{ and } c_\alpha = \frac{1}{\rho_\alpha}\frac{d\rho_\alpha}{dp} \qquad \alpha = w, n, \tag{2.5}$$

respectively. These compressibilities are assumed to be known functions of the pressure.

In this work, we take the wetting phase saturation $S = S_w$ and the total velocity $\mathbf{u} = \mathbf{u}_n + \mathbf{u}_w$ as our primary variables along with the global pressure $p$. Additionally, gravitational forces are neglected.

We may now reformulate eqs. (2.1) to (2.5) as

$$\begin{aligned} \mathbf{u} &= -\lambda_t(S)k\nabla p \\ \phi\left(c_r + (1-S)\,c_n + Sc_w\right)\frac{\partial p}{\partial t} & \\ +\nabla\cdot\mathbf{u} - \left[c_w\lambda_w(S) + c_n\lambda_n(S)\right]\frac{\mathbf{u}\cdot\mathbf{u}}{\lambda_t(S)^2 k} &= q_w + q_n \\ \phi\frac{\partial \rho_w S}{\partial t} + \nabla\cdot\left[\rho_w f_w(S)\mathbf{u}\right] &= \rho_w q_w, \end{aligned} \tag{2.6}$$

where we have introduced the phase mobilities $\lambda_w = k_{r,w}(S)/\mu_w$ and $\lambda_n = k_{r,n}(S)/\mu_n$, the total mobility $\lambda_t = \lambda_n + \lambda_w$, and the fractional flow function $f_w(S) = \lambda_w/\lambda_t$. These equations, along with initial and boundary conditions and the interior pressure jump conditions for the faults which will be introduced in the following section, are what we use as our forward model in this work.

## 2.2 Faults and Fault Transmissibilities

A fault is a heterogeneity characterized by a sharp change in the permeability tensor $\mathbf{k}$ over a region with very small width relative to its length and the overall size of the reservoir domain [1]. Typically, the width of faults is on the order of meters while the size of a reservoir is often on the order of hundreds of meters [9]. Although the presence of a fault can have a significant impact on the flow of fluids in a porous medium, their small width may require a very fine mesh leading to excessively high computational costs [9]. One way to deal with this issue is to consider a reduced fault model in which the fault is treated as a surface of codimension one coupled with the rest of the domain. This avoids the need for an extremely fine grid to resolve the (volumetric) fault.

Here, we assume no flow along the fault so that the fault acts as a barrier to the flow and introduces a discontinuous pressure jump as in [10]. Additionally, the source terms $q_w$ and $q_n$ are both assumed to be identically zero inside the fault. This gives the following jump condition for a fault $\Gamma_i$ in the interior of the domain

$$[[\mathbf{u} \cdot \mathbf{n^i}]]_{\Gamma_i} = 0, \text{ and, } \mathbf{u} \cdot \mathbf{n^i} - t_f^i [[p]]_{\Gamma_i} = 0, \tag{2.7}$$

where $\mathbf{n}^i$ is a vector normal to the fault and $t_f^i$ is the transmissibility multiplier. The transmissibility multiplier characterizes the extent to which fluid can flow across the fault. A low transmissibility means that the flow will tend to avoid the fault in which case the fault is said to be closed. A high value of the transmissibility means that the fault will have negligible impact on the flow and the fault is said to be open. In this work, we are primarily interested in using techniques from uncertainty quantification to estimate the value of fault transmissibilities from data.

With the addition of interior jump conditions for the faults the final system of equations

for a reservoir with $n_f$ faults is

$$\begin{aligned}
\mathbf{u} &= -\lambda_t(S) k \nabla p \\
\phi c \frac{\partial p}{\partial t} + \nabla \cdot \mathbf{u} & \\
- \left[ c_w \lambda_w(S) + c_n \lambda_n(S) \right] \frac{\mathbf{u} \cdot \mathbf{u}}{\lambda_t(S)^2 k} &= q_t \\
\phi \frac{\partial \rho_w S}{\partial t} + \nabla \cdot \left[ \rho_w f_w(S) \mathbf{u} \right] &= \rho_w q_w \\
\left[ [\mathbf{u} \cdot \mathbf{n^i}] \right]_{\Gamma_i} &= 0, \qquad \text{for } i = 1, \ldots, n_f \\
\mathbf{u} \cdot \mathbf{n^i} - t_f^i [[p]]_{\Gamma_i} &= 0, \qquad \text{for } i = 1, \ldots, n_f,
\end{aligned} \tag{2.8}$$

where we have introduced the total compressibility $c = c_r + (1 - S)\, c_n + S c_w$ and $q_t = q_w + q_n$. Along with boundary and initial conditions, eq. (2.8) gives the forward model used in this work.

## 2.3 Discretization

We approximate eq. (2.8) in space by combining a mixed finite element formulation with a DG scheme. The mixed finite element formulation is used to discretize the pressure and velocity equations while we use a DG scheme for the saturation equation. We use lowest-order Raviart-Thomas elements for the velocity and piecewise constant discontinuous elements for the pressure and saturation. The time derivatives are discretized using an implicit Euler method. Although this leads to a nonlinear system, using the implicit Euler method yields more favorable stability than an explicit Euler scheme. This gives the following problem:

Given $S^0$, $p^0$, $\forall j = 1, \cdots, N$ find $(\mathbf{u}^j, p^j, S^j) \in (\mathbf{V_h}, W_h, Z_h)$ such that

$$
\begin{aligned}
\left( \left[ k\lambda_t(S^j) \right]^{-1} \mathbf{u}^j, \mathbf{v} \right) + \sum_{i=1}^{n_f} \langle \frac{1}{t_f^i} \mathbf{u}^j \cdot \mathbf{n}, \mathbf{v} \cdot \mathbf{n} \rangle_{\Gamma_i} & \\
- \left( p^j, \nabla \cdot \mathbf{v} \right) = 0, \qquad & \forall \mathbf{v} \in V_h \\
\left( \phi c \frac{p^j - p^{j-1}}{\Delta t}, w \right) + \left( \nabla \cdot \mathbf{u}^j, w \right) & \\
- \left( \left[ c_w \lambda_w(S^j) + c_n \lambda_n(S^j) \right] \frac{\mathbf{u}^j \cdot \mathbf{u}^j}{\lambda_t^2(S^j) k}, w \right) = (q_t, w), \quad & \forall w \in W_h \\
\left( \phi \frac{\rho_w(p^j) s^j - \rho_w(p^{j-1}) s^{j-1}}{\Delta t}, z \right) & \\
- \left( \rho_w(p^j) f_w(S^j) \mathbf{u}^j, \nabla z \right) + \langle \rho_w(p_{up}^j) f_w(s_{up}^j) \mathbf{u}^j \cdot \mathbf{n}, [z] \rangle_e & \\
- \left( \rho_w(s^j) q_w, z \right) = 0, \qquad & \forall z \in Z_h,
\end{aligned}
\tag{2.9}
$$

where $(\cdot, \cdot)$ and $\langle \cdot, \cdot \rangle$ denote the usual inner products on the volume and edges of the elements of the mesh respectively, $\mathbf{n}$ is the outward unit normal vector field defined on the edges of the mesh, $V_h$ is the space of lowest order Raviart-Thomas basis functions, and $W_h$ and $Z_h$ are the spaces of piecewise constant discontinuous functions defined on the computational mesh. We use $p_{up}^j$ and $s_{up}^j$ to mean the pressure and saturation values on the edges of the mesh chosen to satisy upwinding for the numerical fluxes. We have assumed here that the saturation and pressure on the exterior boundaries are zero so that they do not appear in eq. (2.9). This assumption holds throughout this work.

For the wells, we take the injection wells to have constant pressure and the producers to be rate wells. The water source term $q_w$ is given by

$$
\begin{aligned}
q_w = & \sum_{i=1}^{n_I} \alpha_I^i \lambda_t(S_I^i) J^i \left( p_I^i - p \right) f_w(S_I^i) \phi(x_I^i, \sigma^2) \\
& + \sum_{i=1}^{N_P} -\alpha_P^i Q_P^i f_w(S_P^i) \phi(x_P^i, \sigma^2),
\end{aligned}
\tag{2.10}
$$

where the subscripts $I$ and $P$ denote injection and pressure wells respectively, $x_{I,P}$ are the locations of the wells, $\phi$ is a Gaussian function with standard deviations given by $\sigma$. Here, we take $\sigma \approx 18.768$[1]. The water saturation values at the wells are given by $S_{I,P}$, the term $\alpha_{I,P}$ is a multiplication factor for edge and corner wells, $p_I$ is the constant bottom hole pressure at the injection wells, $Q_P$ is the flow rate at each of the production wells. The multiplication factors $\alpha_{I,P}$ are taken to be 1 for interior wells, 2 for edge wells, and 4 for corner wells. In this work, the only wells considered are corner wells. Finally, $J$ is the well index from Peaceman's model given by

$$J = \frac{2\pi k}{\ln(\frac{r_e}{r_w})}, \tag{2.11}$$

where $r_w$ is the well radius and $r_e$ is an effective radius. In this case, the effective radius is taken to be

$$r_e = \frac{2\sqrt{2}h}{3} e^{-\frac{\pi}{6}}, \tag{2.12}$$

where $h$ is the minimum cell diameter of the computational mesh[2]. Similarly, the source term for the oil phase, $q_n$ is given by

$$\begin{aligned} q_n = &\sum_{i=1}^{n_I} \alpha_I^i \lambda_t(S_I^i) J^i \left(p_I^i - p\right) f_n(S_I^i) \phi(x_I^i, \sigma^2) \\ &+ \sum_{i=1}^{N_P} -\alpha_P^i Q_P^i f_n(S_P^i) \phi(x_P^i, \sigma^2). \end{aligned} \tag{2.13}$$

For a more detailed explanation of the numerical implementation of wells see [4].

We assume that the compressiblities $c_r$, $c_n$, and $c_w$, are constant and that the fluids are slightly compressible so that

$$\rho_\alpha = \rho_{0,\alpha} \left[1 + c_\alpha \left(p - p_0\right)\right], \qquad \alpha = w, n, \tag{2.14}$$

[1]For the interested reader, we take $\sigma$ to be the minimum cell diameter of the computational mesh.
[2]The mesh used in Chapter 5 has minimum cell diameter $h = 18.76820539116728$.

for some reference pressure $p_0$ and reference densities $\rho_{0,w}$ and $\rho_{0,n}$. The rock compressibility in this work is very small $\sim \mathcal{O}(10^{-10})$ so we treat the porosity $\phi$ as constant for convenience.

We solve the system using the FEniCS library [8]. This is an open-source computing platform for solving partial differential equations. The mesh is generated using Gmsh [11] and is conforming to the faults. This means that each of the faults will lie entirely along the facets of the mesh.

# Chapter 3

# Inverse Problems

In this work we investigate uncertainty quantification for inverse problems in petroleum reservoir simulation. This process is known as history matching in the field of petroleum engineering. In this section we describe the formulation of the inverse problem solved in this work. First, we briefly describe history matching, commonly used observational data and inversion targets, and the challenges posed by these types of inverse problems. A more detailed treatment of history matching in reservoirs can be found in [3]. Second, we describe the Bayesian framework for solving inverse problems which is used in this work [12, 13]. Finally, we describe the Markov chain Monte Carlo method which is a common approach to generating samples from the Bayesian posterior in order to estimate moments of the probability distribution [12].

## 3.1 History Matching

History matching is an inverse problem in which observational data collected from a reservoir is used to estimate model parameters that may be difficult to determine through observation alone. Recently, considerable progress has been made in the field of history matching using large amounts of data. This progress can be attributed to an increase in computational power, along with the adoption of Monte Carlo methods [3].

Given a reservoir model $M : X \to D$ and potentially noisy observational data $d_{\text{obs}}$, the

basic problem of history matching is to find the model parameters $x \in X$ such that

$$M(x) = d_{\text{obs}}. \tag{3.1}$$

Due to observational noise and model error, it may be the case that no set of model parameters $x$ exactly matches the observational data $d_{\text{obs}}$ in which case one seeks model parameters $x$ which approximately matches the data. In this case eq. (3.1) can be replaced by a minimization problem, find

$$\text{argmin}_x \ \frac{1}{2}||d_{\text{obs}} - M(x)||_D^2 \tag{3.2}$$

for some norm $||\cdot||_D$ on the space of admisible model ouputs $D$. Additionally, the solution to the history matching problem should include some estimate of the uncertainty in the prediction. Parameters that must be specified in a typical numerical reservoir simulator include porosity, the permeability field, fault transmissibility, intial conditions, etc. In practice, there may be a large number of variables which need to be estimated in a history matching problem. The data $d_{\text{obs}}$ used to estimate the parameters is typically production data taken at the wells. These can be a time series of measurements of flow rate, pressure, or ratios of flow rates at producing or injecting wells. In this work, we consider fault transmissibilities to be the unknown parameters and use measurements of pressure at the wells as our observational data with which to inform the uncertain parameters.

History matching is uniquely challenging for several reasons. First, the relationship between parameters and model outputs is often highly non-linear. Second, although observations may be taken frequently so that the amount of raw data is large, the information content is often quite low. This is due to the limited number of observation locations and the diffusive nature of the flow [3]. Finally, the number of model parameters to estimate may be several orders of magnitude larger than the amount of independent data. The large number of parameters and low information content of the data means that the solution of the inverse

problem is usually ill-posed and must therefore be constrained by prior knowledge. Prior knowledge about the parameters may be included by replacing eq. (3.2) with the regularized minimization problem find

$$\text{argmin}_x \left( \frac{1}{2}||d_{\text{obs}} - M(x)||_D^2 + \frac{1}{2}||x - x_0||_X^2 \right) \tag{3.3}$$

where $||\cdot||_X$ is some norm on the parameter space $X$ and $x_0$ is some set of model parameters. This formulation can be used to eliminate the ill-posedness of eq. (3.2).

## 3.2 The Bayesian Framework

The Bayesian framework for solving inverse problems conveniently addresses in a mathematically rigorous way two of the challenges mentioned in the last section. Namely, a need for quantifying uncertainty in estimated model parameters (rather than obtaining a single set of parameters) and the need for prior knowledge to provide regularization in an ill-posed problem. Rather than solving an optimization problem as in eq. (3.3) the Bayesian approach solves a statistical inference problem. It seeks a probability distribution which updates prior knowledge about the parameters with information from the data which are linked to the parameters through the inverse of the model. The solution to the Bayesian inverse problem is the posterior probability distribution which has the form

$$\pi_{\text{post}}(x|d_{\text{obs}}) \propto \pi_{\text{like}}(d_{\text{obs}}|x)\, \pi_{\text{prior}}(x). \tag{3.4}$$

Where $\pi_{\text{prior}}$ represents the prior beliefs about the parameters and $\pi_{\text{like}}$ is the contribution from the data. Note that eq. (3.4) gives the posterior distribution up to a normalizing constant. In practice, this is often all that is required because one typically generates samples from the posterior using methods which only require evaluating the posterior up to some

normalizing constant. These methods will be discussed in the following section.

The likelihood function $\pi_{\text{like}}$ represents the probability that a given parameter could produce the observed data. Often, one assumes that the observed data $d_{\text{obs}}$ has some noise so that $d_{\text{obs}} = M(x) + \eta$ where $\eta$ is some mean zero random variable distributed according to some known probability density function $\rho$. Then the probability of the observed data $d_{\text{obs}}$ given some parameter $x$ is

$$\pi_{\text{like}}(d_{\text{obs}}|x) = \rho(d_{\text{obs}} - M(x)).$$

Evaluating the likelihood function involves solving the model $M$ which can be expensive if, as in this work, the model involves solving a partial differential equation. The prior $\pi_{\text{prior}}$ is some known probability density function which describes prior beliefs about the parameters. In practice, the prior is often used to add curvature to the posterior and remove ill-posedness from the inverse problem.

## 3.3 Markov Chain Monte Carlo

The Bayesian posterior gives us a theoretical solution to the statistical inverse problem. However, extracting information from the posterior for decision-making can be challenging, particularly if the posterior is high-dimensional and expensive to evaluate which is often the case. Typically, one is interested in moments of the posterior such as the mean and variance to make decisions. In high dimensions, it may be preferable to estimate these moments using Monte Carlo methods involving samples distributed (approximately) according to the posterior rather than quadrature methods for integration. The central limit theorem then guarantees convergence independent of the dimension of the problem [14]. While the error in estimating moments of a distribution using samples is dimension-independent, generating those samples is in general highly dimension-dependent. One popular method for generating

samples from high-dimensional probability distributions are Markov Chain Monte Carlo (MCMC) sampling methods. These methods rely on constructing a Markov chain with some desired stationary distribution, in this case the poseterior $\pi_{\text{post}}$. In this work we will focus on a specific type of commonly used MCMC algorithm known as Metropolis-Hastings (MH) MCMC.

The Metropolis-Hastings algorithm constructs a Markov Chain $\{x_i\}$ by using a proposal distribution $Q$ which depends only on the current state of the chain $x_i$. The algorithm proceeds as follows. A move $y$ is sampled from the probability density function $Q(\cdot; x_i)$. The proposed move $y$ is accepted or rejected with probability given by

$$\alpha(x_i, y) = \min\left\{1, \frac{\pi_{\text{post}}(y)\, Q(x_i; y)}{\pi_{\text{post}}(x_i)\, Q(y; x_i)}\right\}, \tag{3.5}$$

where $\alpha$ is the the acceptance probability. If the proposed move is accepted then $x_{i+1} = y$. Otherwise $x_{i+1} = x_i$.

Together, the proposal and the acceptance probability define a transition kernel $K$. A sufficient condition for a Markov chain to have the posterior $\pi_{\text{post}}$ as its stationary distribution is for it to satisfy the condition of detailed balance with respect to the posterior [12]. The equation of detailed balance is given by

$$\pi_{\text{post}}(x)K(x, y) = \pi_{\text{post}}(y)K(y, x). \tag{3.6}$$

The Metropolis-Hastings algorithm satisfies the condition of detailed balance with respect to the posterior by construction.

In practice, the chain is truncated after a certain number of iterations and the remaining samples are correlated. An important consideration when designing MH-MCMC algorithms is finding a proposal distribution $Q$ which efficiently explores the posterior. Often, the proposal $Q(\cdot; x_i)$ is chosen to be a Gaussian distribution centered at the point $x_i$. In this

case, $Q(\cdot;\cdot)$ is symmetric and eq. (3.5) becomes

$$\alpha(x_i, y) = \min\left\{1, \frac{\pi_{\text{post}}(y)}{\pi_{\text{post}}(x_i)}\right\}. \tag{3.7}$$

The standard deviation of the proposal is known as the step size of the chain. A small step size relative to the spread of the posterior typically leads high acceptance rates. However, if the step size is too small, the samples will be close to one another and the correlation between them will be high. If the step-size is too large, then the samples from $Q$ are likely to be in low-probability regions of the parameter space and will be rejected leading to many wasted model solves.

# Chapter 4

# Methodology

The goal of this Chapter is to develop the framework for a sequential Metropolis-Hastings transition kernel for MCMC when the target distribution is a posterior with likelihood defined by time-series data. First, the Bayesian inverse problem with time-series data is described and the likelihood is factored into a product of distributions which represent the contribution of the observed data from distinct times. Second, the sequential transition kernel is described. This is done by first outlining a naive approach to a sequential transition kernel which illustrates the basic concept, then building to the sequential transition kernel used in this work. Finally, the sequential transition kernel is applied to an artificial problem and compared with the standard Metropolis-Hastings approach.

## 4.1 Problem Setup

Consider a time-dependent parameter-to-observable map $\mathcal{M} : X \to D$ and noisy time-series data $d = \mathcal{M}(x_{\text{true}}) + \eta$ where $x_{\text{true}}$ is the unknown true parameter and $\eta$ is a mean-zero Gaussian random variable so that $\eta \sim \mathcal{N}(0, \Sigma)$. The data likelihood is then $\rho(d|x) = \mathcal{N}(d - \mathcal{M}(x), \Sigma)$. Given some prior $\pi_{\text{prior}}$ the posterior is

$$\pi_{\text{post}}(x) \propto \pi_{\text{prior}}(x)\,\rho(d|x) = \pi_{\text{prior}}(x)\,\mathcal{N}(d - \mathcal{M}(x), \Sigma)$$

Now let $\mathcal{M}_t(x)$ denote the parameter-to-observable map for a single time-step $t$ and let $d_t$ denote the subset of data for that time-step. There may be more than one observation at a single time-step, for example if there are observations at multiple spatial locations. We assume that observational noise is uncorrelated in time although there may be correlation between observational noise within a single time-step. This means that $\Sigma$ is at least block diagonal and that the likelihood can be factored into a product of lower dimensional Gaussian probability density functions

$$\rho(d|x) = \mathcal{N}(d - \mathcal{M}(x), \Sigma) = \prod_{t=1}^{T} \mathcal{N}(d_t - \mathcal{M}_t(x), \Sigma_t)$$

where $\Sigma_t$ is the block of the covariance matrix $\Sigma$ corresponding to time-step $t$. Each of these probability density functions describes the contribution of the data collected at the corresponding time-step to the overall posterior. Using this factorization of the likelihood, the posterior can be factored as well

$$\pi_{\text{post}}(x) \propto \pi_{\text{prior}}(x) \prod_{t=1}^{T} \mathcal{N}(d_t - \mathcal{M}_t(x), \Sigma_t). \tag{4.1}$$

## 4.2 The Sequential Transition Kernel

In this section, the sequential transition kernel for generating samples from the posterior described in eq. (4.1) is presented. The concept is first illustrated with a simple implementation which lacks efficiency but is instructive. A more sophisticated sequential transition kernel is then developed which has increased efficiency but may not satisfy the principle of detailed balance with respect to the posterior which is a sufficient condition for the transition kernel to have the posterior $\pi_{\text{post}}$ as a stationary distribution. Finally, the sequential transition kernel used in this work is described.

### 4.2.1 Naive Sequential Transition Kernel

Consider a standard Metropolis-Hastings MCMC algorithm in which the goal is to generate samples from the posterior in eq. (4.1). Given the current position of the chain $X(i) = x$, a move $y$ is simulated from a proposal distribution $Q(\cdot; x)$. This move is accepted with probability

$$\min\left(1, \frac{\pi_{\text{post}}(y)}{\pi_{\text{post}}(x)} \frac{Q(x; y)}{Q(y; x)}\right).$$

If the move is accepted then $X(i+1) = y$. Otherwise, $X(i+1) = x$. We assume here that the proposal distribution $Q(\cdot; x)$ is a Gaussian probability density centered at $x$ such that $Q(x; y) = Q(y; x)$. In this case the acceptance probability simplifies to the ratio of the posterior evaluated at the proposed move $y$ and the current position of the chain $x$. We denote this ratio by $r = \frac{\pi_{\text{post}}(y)}{\pi_{\text{post}}(x)}$ and have that the acceptance probability is

$$\min\left(1, \frac{\pi_{\text{post}}(y)}{\pi_{\text{post}}(x)}\right) = \min(1, r).$$

Using the decomposition of the posterior into separate time-steps we may write the ratio $r$ as

$$\begin{aligned} r = \frac{\pi_{\text{post}}(y)}{\pi_{\text{post}}(x)} &= \frac{\pi_{\text{prior}}(y) \prod_{t=1}^{T} \mathcal{N}(d_t - \mathcal{M}_t(y), \Sigma_t)}{\pi_{\text{prior}}(x) \prod_{t=1}^{T} \mathcal{N}(d_t - \mathcal{M}_t(x), \Sigma_t)} \\ &= \frac{\pi_{\text{prior}}(y)}{\pi_{\text{prior}}(x)} \prod_{t=1}^{T} \frac{\mathcal{N}(d_t - \mathcal{M}_t(y), \Sigma_t)}{\mathcal{N}(d_t - \mathcal{M}_t(x), \Sigma_t)}. \end{aligned}$$

Let $r_t = \frac{\mathcal{N}(d_t - \mathcal{M}_t(y), \Sigma_t)}{\mathcal{N}(d_t - \mathcal{M}_t(x), \Sigma_t)}$ for $t = 1, \cdots, T$ and let $r_0 = \frac{\pi_{\text{prior}}(y)}{\pi_{\text{prior}}(x)}$. Throughout this section, we will refer to $r$ and $r_t$ as defined above for a current position of the chain $x$ and proposed move $y$.

We denote the standard Metropolis-Hastings acceptance probability by $\alpha(x, y)$ and can

write this accpetance probability as

$$\alpha(x, y) = \min\left(1, \prod_{t=0}^{T} r_t\right).$$

For now, assume without loss of generality that the contribution of the prior $r_0$ is negligible (as in the case of a very broad prior) so that

$$\alpha(x, y) = \min\left(1, \prod_{t=1}^{T} r_t\right). \tag{4.2}$$

At the end of this section, we will discuss how to incorporate the prior into the sequential transition kernel.

Notice that computing the ratio $r$ requires solving the model for all $T$ time-steps. This could be computationally expensive if the model is a PDE with a large number of time-steps. In many practical problems including reservoir simulation, the information content of time series data is often fairly low [3]. In other words, observations in time often provide redundant (or nearly redundant) information. Mathematically, this means that blocks of the Jacobian of the parameter-to-observable map corresponding to different time-steps span nearly the same space. This suggests that one could use model outputs from initial time-steps to make decisions about whether or not to reject a proposed move without having to compute the full likelihood.

To be more precise, consider the special case in which $r_t < 1 \ \forall\, t$. Rather than computing the full likelihood for the proposed jump, one could evaluate $r_1$ and reject the move with probability $1 - r_1$. This process is repeated for $t = 1, \cdots, T$ and if the jump is not rejected at any time-step then it is accepted and added to the chain. If the move is rejected at any point, then $X(t + 1) = x$ and no more model solves are performed. If the proposed move is not rejected at any time-step $t$, then $X(t + 1) = y$. After $y$ has been either accepted or

rejected, a new move is sampled from the proposal and the process is repeated until the desired number of samples have been generated. The probability of not rejecting a move at a time-step $t$ is $\min(1, r_t)$ which, using the assumption that $r_t < 1$, is simply $r_t$. Since a move is accepted if and only if it is not rejected at each time-step, the probability of accepting a move is $\prod_{t=1}^{T} r_t = r$ so that this naive sequential acceptance probability has the same probability of accepting as that of a standard MH-MCMC acceptance probability (again using the fact that each $r_t < 1$). Thus, in the special case for which $r_t < 1$ for all $t$, this naive sequential acceptance probability results in the same probability of accepting with potentially fewer model solves. Similarly, in the case for which $r_t > 1 \ \forall\, t$, the sequential acceptance probability has the same probability of accepting as the standard Metropolis-Hastings acceptance probability. In this case, the proposed move will not be rejected at any time-step since $(1 - r_t) < 0$ and so the move is accepted with probability 1 as it would be in the standard Metropolis-Hastings acceptance probability. We denote the acceptance probability for this naive algorithm by $\alpha^{\text{naive}}(x, y)$. In general, without any assumptions on the values $r_t$, it accepts with probability given by

$$\alpha^{\text{naive}}(x, y) = \prod_{t=1}^{T} \min\{1, r_t\}. \tag{4.3}$$

Together, the proposal $Q$ and acceptance probability $\alpha^{\text{naive}}$ define the naive sequential transition kernel. An implementation of this naive approach to a sequential transition kernel is shown in Algorithm 1.

**Proposition 4.2.1.** *The naive sequential acceptance probability $\alpha^{naive}$ is less than or equal to the standard Metropolis-Hastings acceptance probability $\alpha$. That is, $\forall\, x, y \in \mathcal{X}$,*

$$\alpha^{naive}(x, y) \leq \alpha(x, y).$$

*Proof.* Clearly, $\min\{1, r_t\} \leq 1,\ \forall\, t \in \{1, \cdots, T\}$ which implies that

$$\alpha^{\text{naive}}(x, y) = \prod_{t=1}^{T} \min\{1, r_t\} \leq 1$$

Similarly, $\min\{1, r_t\} \leq r_t,\ \forall\, t \in \{1, \cdots, T\}$ which implies that

$$\alpha^{\text{naive}}(x, y) = \prod_{t=1}^{T} \min\{1, r_t\} \leq \min\{1, \prod_{t=1}^{T} r_t\} = \alpha(x, y)$$

□

**Algorithm 1** Naive Sequential Metropolis-Hastings Transition Kernel

Given current position $x$ and proposed move $y$

**for** $t = 1, \cdots, T$ **do**
    Evaluate $r_t$ by computing a single-time step of the model
    Draw $\eta \sim \mathcal{U}(0, 1)$
    **if** $r_t < \eta$ **then**
        Reject $y$
        break
    **else**
        continue
    **end if**
**end for**

### 4.2.2 A More Sophisticated Sequential Transition Kernel

The naive transition kernel described in the previous section illustrates how a sequential transition kernel can be implemented for MCMC. However, it may have a lower acceptance probability than the standard approach. This will occur when the $r_t$ are not uniformly less than or greater than 1. In this section a new algorithm for a sequential transition kernel which has potentially larger acceptance probability than the naive approach is developed. Consider the simple case in which $T = 2$ and $r_1 = 4$ while $r_2 = 0.5$. In a standard Metropolis-Hastings

approach, the acceptance probability would be

$$\alpha(x, y) = \min\{1, r_1\, r_2\} = 1.$$

If we were to apply Algorithm 1 to this example, then the probability of accepting this move would be

$$\alpha^{\text{naive}}(x, y) = \min\{1, r_1\} * \min\{1, r_2\} = 0.5.$$

Intuitively, when initial time-steps result in values of the likelihood ratios $r_t > 1$, this information should propagate to later time-steps in some way. After computing $r_2$, it doesn't make sense to reject with probability $1 - r_2 = 0.5$ since we also have access to the value of $r_1$ and we know that $r_1\ r_2 = 2$. This suggests that at step $t$ in the sequential approach, rather than only using $r_t$ and rejecting with probability $1 - r_t$, all information up to that point should be used and a move should be rejected with probability $1 - \prod_{i=1}^{t} r_i$ at each step $t$. However, since a rejection stage is performed whenever the product of the likelihood ratios is less than 1, this only makes sense as long as $\prod_{i=1}^{t-1} r_i \geq 1$. If we consider the reverse jump so that $r_1 = 0.25$ and $r_2 = 2$ then rejecting at each step with probability $1 - \prod_{i=1}^{t} r_i$ results in an acceptance rate of $\min\{1, r_1\} * \min\{1, r_1\ r_2\} = r_1^2\ r_2 = 0.125$. This scheme penalizes the proposed move twice for the low value of $r_1$. The fix for this problem is to define $p_t$ at each time-step where

$$\begin{aligned} &p_1 = r_1, \\ &p_t = p_{t-1}\ r_t \qquad \text{if } p_{t-1} > 1, \\ &p_t = r_t \qquad \text{otherwise} \end{aligned} \tag{4.4}$$

and reject at each step $t$ with probability $1 - p_t$. We denote this new acceptance probability by $\hat{\alpha}^{\text{seq}}(x, y)$ and it is given by

$$\hat{\alpha}^{\text{seq}}(x, y) = \prod_{t=1}^{T} \min\{1, p_t\}. \tag{4.5}$$

Using this new approach, the acceptance probabilities for the previous example are now $\hat{\alpha}^{\text{seq}}(x, y) = 1$ and $\hat{\alpha}^{\text{seq}}(y, x) = 0.25$. The acceptance probability $\hat{\alpha}^{\text{seq}}$, with the proposal $Q$ defines a new transition kernel $\hat{K}^{\text{seq}}$. This transition kernel is implemented in Algorithm 2. The transition kernel in Algorithm 2 results in acceptance probabilities which are larger than those in the naive implementation in Algorithm 1 and closer to the acceptance probabilities in a standard MH algorithm.

**Proposition 4.2.2.** *The acceptance probability $\hat{\alpha}^{seq}$ is greater than or equal to the naive sequential acceptance probability $\alpha^{naive}$. That is, $\forall\, x, y \in \mathcal{X}$,*

$$\alpha^{naive}(x, y) \leq \hat{\alpha}^{seq}(x, y).$$

*Proof.* Clearly, $r_t \leq p_t,\ \forall\, t$ by definition of $p_t$. It follows that

$$\alpha^{\text{naive}}(x, y) = \prod_{t=1}^{T} \min\{1, r_t\} \leq \prod_{t=1}^{T} \min\{1, p_t\} = \hat{\alpha}^{\text{seq}}(x, y)$$

□

**Proposition 4.2.3.** *The sequential acceptance probability $\hat{\alpha}^{seq}$ is less than or equal to the standard Metropolis-Hastings acceptance probability $\alpha$. That is, $\forall\, x, y \in \mathcal{X}$,*

$$\hat{\alpha}^{seq}(x, y) \leq \alpha(x, y).$$

*Proof.* We need to show that

$$\prod_{t=1}^{T} \min\{1, p_t\} \leq \min\{1, \prod_{t=1}^{T} r_t\}$$

Clearly,

$$\prod_{t=1}^{T} \min\{1, p_t\} \leq 1$$

so, without loss of generality, assume that $\prod_{t=1}^{T} r_t < 1$.

Let $\{t_i\}_{i=1}^{N}$ be the set of all indices such that $p_{t_i} < 1$ and $t_i < t_{i+1}$ for $i = 1, \cdots, N-1$. The assumption that $\prod_{t=1}^{T} r_t < 1$ implies that this set is non-empty. To see this, suppose to the contrary that $\{t_i\}_{i=1}^{N} = \emptyset$. Then by induction $p_T = \prod_{t=1}^{T} r_t$. Now by assumption, $\prod_{t=1}^{T} r_t < 1$ which implies that $T \in \{t_i\}_{i=1}^{N}$ and contradicts the assumption that $\{t_i\}_{i=1}^{N} = \emptyset$.

Now, $p_{t_1} = \prod_{t=1}^{t_1} r_t$ by induction and the fact that $p_t > 1 \ \forall t < t_1$. This also means that $\prod_{t=1}^{t_1} \min\{1, p_t\} = p_{t_1}$. Together, these two equalities show that

$$\prod_{t=1}^{t_1} \min\{1, p_t\} = p_{t_1} = \prod_{t=1}^{t_1} r_t.$$

If $N \geq 2$, a similar argument shows that

$$\prod_{t=t_i+1}^{t_{i+1}} \min\{1, p_t\} = p_{t_{i+1}} = \prod_{t=t_i+1}^{t_{i+1}} r_t,$$

for $i = 1, \cdots, N-1$. So now we have that

$$\prod_{t=1}^{t_N} \min\{1, p_t\} = \prod_{t=1}^{t_N} r_t,$$

If $t_N = T$ then we are done. If not, that is $t_N < T$, then it remains to show that

$$\prod_{t=t_N+1}^{T} \min\{1, p_t\} < \prod_{t=t_N+1}^{T} r_t.$$

Since $p_t > 1 \; \forall t > t_N$ we have that $\prod_{t=t_N+1}^{T} \min\{1, p_t\} = 1$ and by induction that $p_T = \prod_{t=t_N+1}^{T} r_t$. Using the assumption that $p_T > 1$ we have that

$$\prod_{t_N+1}^{T} \min\{1, p_t\} = 1 < \prod_{t=t_N+1}^{T} r_t.$$

This completes the proof.

□

**Algorithm 2** Sequential Metropolis-Hastings Transition Kernel

Given current position $x$ and proposed move $y$

Set $p_0 = 1$
**for** $t = 1, \cdots, T$ **do**
    Evaluate $r_t$ by computing a single-time step of the model
    **if** $p_{t-1} > 1$ **then**
        $p_t = p_{t-1} * r_t$
    **else**
        $p_t = r_t$
    **end if**
    Draw $\eta \sim \mathcal{U}(0, 1)$
    **if** $r_t < \eta$ **then**
        Reject $y$
        break
    **end if**
**end for**

### 4.2.3 The Full Sequential Transition Kernel

The transition kernel $\hat{K}^{\text{seq}}$ described in the previous section may have larger acceptance probabilities than the naive implementation $K^{\text{naive}}$ but also may no longer satisfy the princi-

ple of detailed balance with respect to the posterior $\pi_{\text{post}}$. This means that $\exists\ x, y \in \mathcal{X}$ such that

$$\pi_{\text{post}}(x)\,\hat{K}^{\text{seq}}(x,y) \neq \pi_{\text{post}}(y)\,\hat{K}^{\text{seq}}(y,x)$$

Returning to the example above in which a proposed jump from $x$ to $y$ results in likelihood ratios of $r_1 = 4$ and $r_2 = 0.5$, the sequential transition kernel described in Algorithm 2 will result in the same acceptance probability as the standard Metropolis-Hastings acceptance probability. However, when considering the reverse jump from $y$ to $x$, we have that $r_1 = 0.25$ and $r_2 = 2$. In this case, the transition kernel described in Algorithm 2 will accept with probability 0.25 while the standard MH transition kernel results in an acceptance probability of 0.5. This illustrates a fundamental difficulty with the sequential transition kernel: when likelihood ratios $r_t$ are less than one for initial time-steps but greater than one at future time-steps the sequential transition kernel may accept with a lower probability than the Metropolis-Hastings transition kernel. The MH transition kernel satisfies detailed balance with respect to the posterior:

$$\pi_{\text{post}}(x)\,\alpha(x,y)\,Q(y;x) = \pi_{\text{post}}(y)\,\alpha(y,x)\,Q(x;y), \qquad x \neq y.$$

In the above examples $\hat{\alpha}^{\text{seq}}(x,y) = 1 = \alpha(x,y)$ but $\hat{\alpha}^{\text{seq}}(y,x) = 0.25 < 0.5 = \alpha(y,x)$ which demonstrates that the sequential transition kernel $\hat{K}^{\text{seq}}$ may not satisfy the detailed balance principle with respect to the posterior. A simple way to enforce the detailed balance principle for the sequential transition kernel is to ensure that

$$\frac{\hat{\alpha}^{\text{seq}}(x,y)}{\alpha(x,y)} = \frac{\hat{\alpha}^{\text{seq}}(y,x)}{\alpha(y,x)}.$$

This can be done by adding an additional rejection stage to the sequential approach. Let $S(x,y)$ denote the ratio of the sequential acceptance probability to the standard MH accep-

tance probability so that

$$S(x,y) = \frac{\hat{\alpha}^{\text{seq}}(x,y)}{\alpha(x,y)}. \tag{4.6}$$

If a proposed jump $y$ from the current position of the chain $x$ is not rejected at any step in Algorithm 2, the transition probabilities for all four quantities above are computed (note that this does not require any additional model solves as we have solved the model for all time-steps for both parameters $x$ and $y$). The move is then rejected with probability

$$1 - \frac{S(y,x)}{S(x,y)}.$$

This finally gives us the sequential transition kernel used in this work. We denote the sequential acceptance probability as $\alpha^{\text{seq}}(x,y)$ and it is given by

$$\alpha^{\text{seq}}(x,y) = \hat{\alpha}^{\text{seq}}(x,y) \min\left\{1, \frac{S(y,x)}{S(x,y)}\right\}. \tag{4.7}$$

An implementation of this transition kernel is shown in Algorithm 3.

**Proposition 4.2.4.** *The sequential transition kernel $K^{seq}$ satifies the principle of detailed balance with respect to the posterior $\pi_{post}$.*

*Proof.* We need to show that

$$\pi_{\text{post}}(x)\alpha^{\text{seq}}(x,y)Q(y;x) = \pi_{\text{post}}(y)\alpha^{\text{seq}}(y,x), Q(x;y) \qquad x \neq y.$$

Using eq. (4.7), this is equivalent to showing that

$$\pi_{\text{post}}(x)\hat{\alpha}^{\text{seq}}(x,y)\min\left\{1, \frac{S(y,x)}{S(x,y)}\right\} Q(y;x) = \pi_{\text{post}}(y)\hat{\alpha}^{\text{seq}}(y,x)\min\left\{1, \frac{S(x,y)}{S(y,x)}\right\} Q(x;y).$$

Now without loss of generality, assume that $S(y,x) \geq S(x,y)$ so that we have

$$\pi_{\text{post}}(x)\hat{\alpha}^{\text{seq}}(x,y)Q(y;x) = \pi_{\text{post}}(y)\hat{\alpha}^{\text{seq}}(y,x)\frac{S(x,y)}{S(y,x)}Q(x;y).$$

Now using eq. (4.6) this becomes

$$\pi_{\text{post}}(x)\hat{\alpha}^{\text{seq}}(x,y)Q(y;x) = \pi_{\text{post}}(y)\hat{\alpha}^{\text{seq}}(y,x)\frac{\left[\frac{\hat{\alpha}^{\text{seq}}(x,y)}{\alpha(x,y)}\right]}{\left[\frac{\hat{\alpha}^{\text{seq}}(y,x)}{\alpha(y,x)}\right]}Q(x;y).$$

Dividing both sides by the quantity $\left[\frac{\hat{\alpha}^{\text{seq}}(x,y)}{\alpha(x,y)}\right]$ gives

$$\pi_{\text{post}}(x)\hat{\alpha}^{\text{seq}}(x,y)\left[\frac{\hat{\alpha}^{\text{seq}}(x,y)}{\alpha(x,y)}\right]^{-1}Q(y;x) = \pi_{\text{post}}(y)\hat{\alpha}^{\text{seq}}(y,x)\left[\frac{\hat{\alpha}^{\text{seq}}(y,x)}{\alpha(y,x)}\right]^{-1}Q(x;y)$$

which is equivalent to

$$\pi_{\text{post}}(x)\alpha(x,y)Q(y;x) = \pi_{\text{post}}(y)\alpha(y,x)Q(x;y).$$

This is the principle of detailed balance for Metropolis-Hastings when $x \neq y$ which is known to be true.

Finally, if $x = y$ then the equation of detailed balance is satisfied trivially. □

**Algorithm 3** Sequential Metropolis-Hastings Transition Kernel

Given current position $x$ and proposed move $y$

Set $p_0 = 1$
accept=True
**for** $t = 1, \cdots, T$ **do**
    Evaluate $r_t$ by computing a single-time step of the model
    **if** $p_{t-1} > 1$ **then**
        $p_t = p_{t-1} * r_t$
    **else**
        $p_t = r_t$
    **end if**
    Draw $\eta \sim \mathcal{U}(0, 1)$
    **if** $r_t < \eta$ **then**
        accept=False
        break
    **end if**
**end for**
**if** accept=True **then**
    Compute $S(x, y)$ and $S(y, x)$
    Draw $\eta \sim \mathcal{U}(0, 1)$
    **if** $\frac{S(y,x)}{S(x,y)} < \eta$ **then**
        accept=False
    **end if**
**end if**

### 4.2.4 Including the Prior in the Sequential Transition Kernel

At the beginning of this section we assumed that the contribution of the prior $r_0$ to the ratio of the posterior values of the parameters $x$ and $y$ was negligible. Relaxing that assumption, there are several ways in which the prior can be included in the sequential transition kernel. A natural approach, and the one taken here, is to simply treat the prior just like the likelihood ratios resulting from model evalutations. In this case, one simply needs to add an extra iteration to the for-loop in the sequential algorithm described above. One could incorporate the prior in other ways such as including it at the end of the sequential algorithm.

## 4.3 Example

In the previous section we described an implementation of a sequential transition kernel for MCMC when the target is a Bayesian posterior resulting from a time-dependent model and independent (in time) observations with Gaussian noise. In this section, we apply this method to an artificial problem and compare its performance to a standard MH-MCMC approach which ignores the sequential nature of the model. The choice of step-size in the proposal distribution is important for performance in both methods. We show in this example that as the step size increases, the sequential approach tends to be more efficient than the standard MH approach.

### 4.3.1 Inverse Problem Setup

The simplest possible model with which to demonstrate the sequential approach is the identity map. In this case we take our model to be $\mathcal{M} : \mathbb{R} \to \mathbb{R}^T$ where

$$\mathcal{M}_t(x) = x, \qquad \text{for } t = 1, \cdots, T. \tag{4.8}$$

Although this model is exceedingly simple, it will demonstrate when one might expect the sequential transition kernel outlined above to be more efficient that the standard MH approach. In this case, we choose $T = 100$ and we assume that the prior is so broad that its effects are negligible so that we only consider the likelihood of the posterior. We take $x_{\text{true}} = 0$ and generate synthetic noisy data by corrupting the true data with Gaussian noise. The observations are then given by

$$d_{\text{obs}} = \mathcal{M}\left(x_{\text{true}}\right) + \mathcal{N}\left(0, \sigma^2 \mathbb{I}_{100}\right), \tag{4.9}$$

where $\mathbb{I}_{100}$ is the $100 \times 100$ identity matrix and the noise is $\sigma = 0.5$. The likelihood is

$$\pi_{\text{like}} = \prod_{i=1}^{100} \mathcal{N}\left(d^i_{\text{obs}}, 0.5^2\right), \tag{4.10}$$

where $d^i_{\text{obs}}$ is the $i$-th datum. The posterior is the product of Gaussians and is itself a Gaussian with mean $\mu = \frac{1}{100}\sum_{i=1}^{100} d^i_{\text{obs}}$ equal to the average of the observations and variance $\sigma^2_{\text{post}} = \frac{\sigma^2}{100}$.

$$\pi_{\text{post}}(x) = \mathcal{N}\left(\mu_{\text{post}}, \sigma^2_{\text{post}}\right). \tag{4.11}$$

### 4.3.2 Autocorrelation and Measuring Efficiency

We have shown above that the sequential approach satisfies detailed balance with respect to the posterior. This means that a Markov chain using the sequential transition kernel is guaranteed to have the posterior as a stationary distribution. In this example we are interested in comparing the relative performance of the sequential approach to MH MCMC for this simple problem. Typically, the purpose of running an MCMC algorithm is to generate a set of samples which are distributed approximately according to the posterior. These samples can then be averaged to estimate moments of the posterior. When the samples are independent, the variance in the estimate decreases at a rate of $1/N$. However, the samples obtained from MCMC algorithms are not independent. The variance in the estimate obtained by averaging over samples from MCMC algorithms decreases at a rate of $\tau/N$ where $\tau$ is the integrated autocorrelation time given by

$$\tau = 1 + 2\sum_{s=1}^{\infty} \rho(s), \tag{4.12}$$

and $\rho(s)$ is the autocorrelation function for a given lag s [12]. The quantity $N/\tau$ is called the effective sample size of the chain. The integrated autocorrelation time is defined for a

scalar-valued function of the chain. In other words, for any moment of the chain we wish to compute, we must determine an integrated autocorrelation time for that moment specifically. In this work, we only consider the integrated autocorrelation times for the mean of a chain. The example considered here has a single parameter dimension, so we consider only a single integrated autocorrelation time.

In practice, Markov chains from MCMC algorithms are finite so the sum in eq. (4.12) is truncated for some $s_0 << N$ where $N$ is the length of the chain. We use the autocorrelation function from the *statsmodels* python module [15]. The autocorrelation function in this library also returns confidence bounds on the autocorrelation function $\rho(s)$ for each lag $s$ computed according to Bartlett's formula. We compute 95% confidence bounds for the autocorrelation function $\rho(s)$ for each lag $s$. We use these confidence bounds in the computation of the integrated autocorrelation time in order to determine reasonable confidence bounds on $\tau$.

For a given proposal distribution, the sequential approach will require fewer model solves than the standard MH approach but will have larger integrated autocorrelation $\tau$ since the sequential transition kernel has strictly lower acceptance probability than the MH transition kernel. We determine the relative efficiency of the sequential approach to the MH approach as follows. Given a set of data $d_{\text{obs}}$ generated as described above, we run the sequential and MH approaches for $N_{seq}$ and $N_{MH}$ iterations respectively on the resulting posterior eq. (4.11). We then determine the average number of model solves required per sample by each method denoted by $n^{seq}_{solves}$ and $n^{MH}_{solves}$. Since we are considering individual time-steps of our model in the sequential approach the term model solves is somewhat ambiguous. Here, we use the term model solve to mean computing one time-step of the model. So in this example, evaluating the transition kernel for MH-MCMC requires 100 model solves per iteration while for the sequential approach it will vary. Finally, we use eq. (4.12) with the summation truncated at some $s_0 << N$, to estimate $\tau_{seq}$ and $\tau_{MH}$, the integrated autocorrelation values for the

sequential and MH approaches, respectively. We then measure the efficiency of the sequential approach relative to MH by the ratio

$$\epsilon = \frac{n_{solves}^{MH}\,\tau_{MH}}{n_{solves}^{seq}\,\tau_{seq}}. \tag{4.13}$$

The quantity $\epsilon$ in eq. (4.13) is the efficiency of the sequential approach relative to MH-MCMC. It can be thought of as the inverse of the fraction of model solves required by the sequential approach to generate the same effective sample size as the MH approach. To see this, consider a chain generated using MH-MCMC which has effective sample size given by $N_{MH}/\tau_{MH}$ and suppose we use the sequential approach to generate a chain which has the same effective sample size so that

$$\begin{aligned} \frac{N_{seq}}{\tau_{seq}} &= \frac{N_{MH}}{\tau_{MH}} \\ \iff N_{seq} &= N_{MH}\frac{\tau_{seq}}{\tau_{MH}} \end{aligned} \tag{4.14}$$

Now the number of total model solves required by the sequential approach is $n_{solves}^{seq}\,N_{seq}$ and combining this with eq. (4.14) we have that

$$\begin{aligned} n_{solves}^{seq}\,N_{seq} &= n_{solves}^{seq}\,N_{MH}\,\frac{\tau_{seq}}{\tau_{MH}} \\ &= n_{solves}^{MH}\,N_{MH}\,\frac{n_{solves}^{seq}\,\tau_{seq}}{n_{solves}^{MH}\,\tau_{MH}} \\ &= n_{solves}^{MH}\,N_{MH}\,\frac{1}{\epsilon}, \end{aligned} \tag{4.15}$$

which shows that the number of model solves required by the sequential approach to generate the same effective sample size as MH-MCMC is $\frac{1}{\epsilon}$ times the total number of solves as the MH-MCMC approach.

### 4.3.3 MCMC Results

The efficiency, as defined in eq. (4.13) depends on the step-sizes of the chains. It is known that the optimal step size for MH-MCMC when the target is a univariate Gaussians is $\eta_{\text{opt}} \approx 2.4\,\sigma_{\text{post}}$ where $\sigma_{\text{post}}$ is the standard deviation of the target which is in this case the posterior defined by eq. (4.11). We compare the efficiency of the sequential and MH approaches using proposal distributions with various step-sizes scaled by the optimal step size $\eta_{\text{opt}}$. In particular, we apply both approaches to the inverse problem described above for step sizes of 0.5, 1, 2, 3, 4, 5, 8, and 10, each scaled by $\eta_{\text{opt}}$. We run both the sequential and MH chains for $10^6$ iterations for each step size. No burn-in period was used because both chains were initialized at the true parameter $x_{\text{true}}$. This was repeated 5 times for different sets of data $d_{\text{obs}}$ in order to determine the average behavior of each method for this problem.

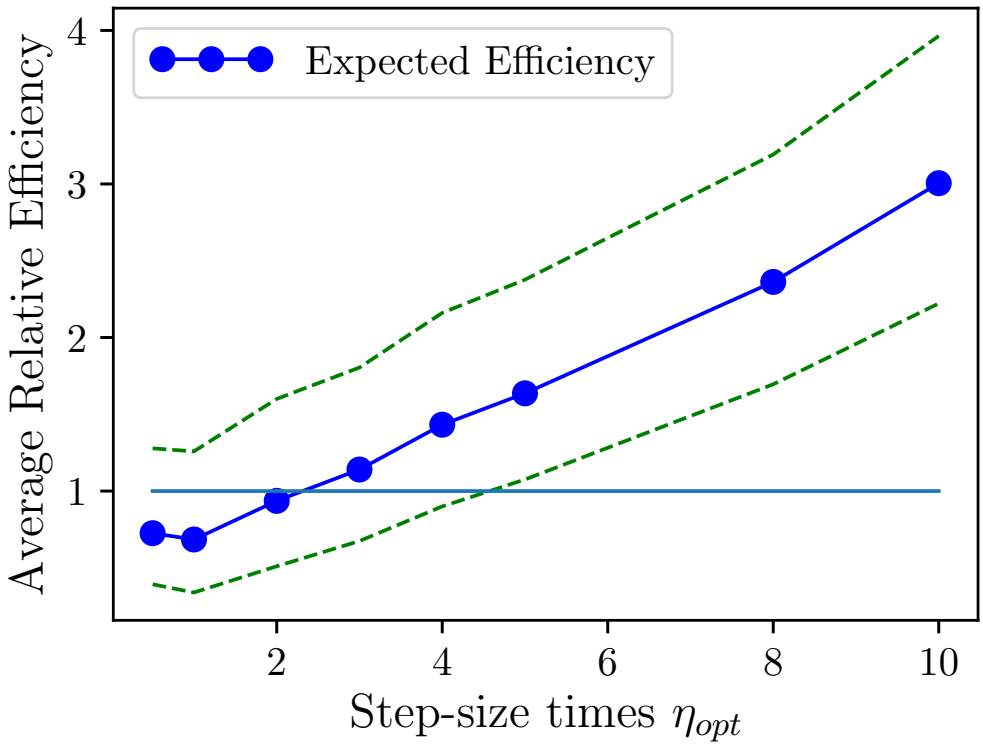



Figure 4.1: Plot of efficiency for various step-sizes using 400 lags in the integrated autocorrelation time.

In Figure 4.1 we show the efficiency of the sequential method relative to the MH method for various step sizes each scaled by the optimal step-size $\eta_{\text{opt}}$. The blue dots represent the efficiency as defined in eq. (4.13) averaged over the five experiments. We use $s_0 = 400$ lags to compute the integrated autocorrelation times for both methods. Clearly, the relative performance of the sequential approach increases with step-size. At the optimal step-size, the MH approach outperforms the sequential method. However, as the step-size increases to $10\,\eta_{\text{opt}}$ we see that the sequential approach begins to outperform the MH approach. The autocorrelation function $\rho$ is a noisy function of the lag. We compute 95% confidence bounds for the autocorrelation function for each lag $s$ in both methods. The dotted lines in Figure 4.1 show the efficiency when these bounds are used in the sum in eq. (4.12). The bottom dotted

line shows the efficiency when the upper bounds are used in the computation of $\tau_{seq}$ and lower bounds are used in the computation of $\tau_{MH}$ while the top dotted line shows the reverse. Again, these quantities are averaged over all five experiments with different sets of data $d_{\text{obs}}$. These lines give us an idea of the range of the efficiency due to noise in the autocorrelation function. The quantity $n_{solves}^{seq}$ is also a noisy function but it steadies out very quickly as the number of samples in the chain increases. In general, the noise in the efficiency is dominated by the noise from the autocorrelation times for both methods.

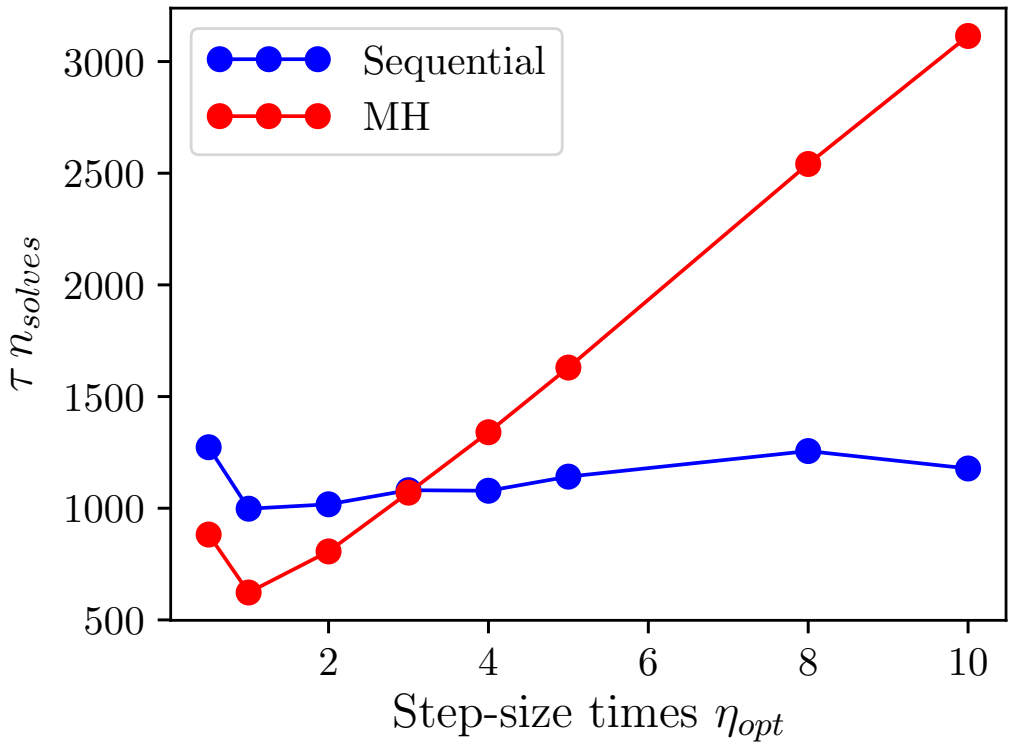



Figure 4.2: Model solves times integrated autocorrelation with 400 lags for the sequential and MH-MCMC approaches for various step-sizes scaled by $\eta_{\text{opt}}$.

In Figure 4.2 we show the quantity $\tau\, n_{solves}$ for both approaches using different step-sizes. As with efficiency, the values shown are averaged over all experiments. We see that the reason for the increase in efficiency in the sequential approach is that the MH-MCMC method becomes less efficient as the step-size increases (as expected), while the sequential method is less sensitive to larger step-sizes.

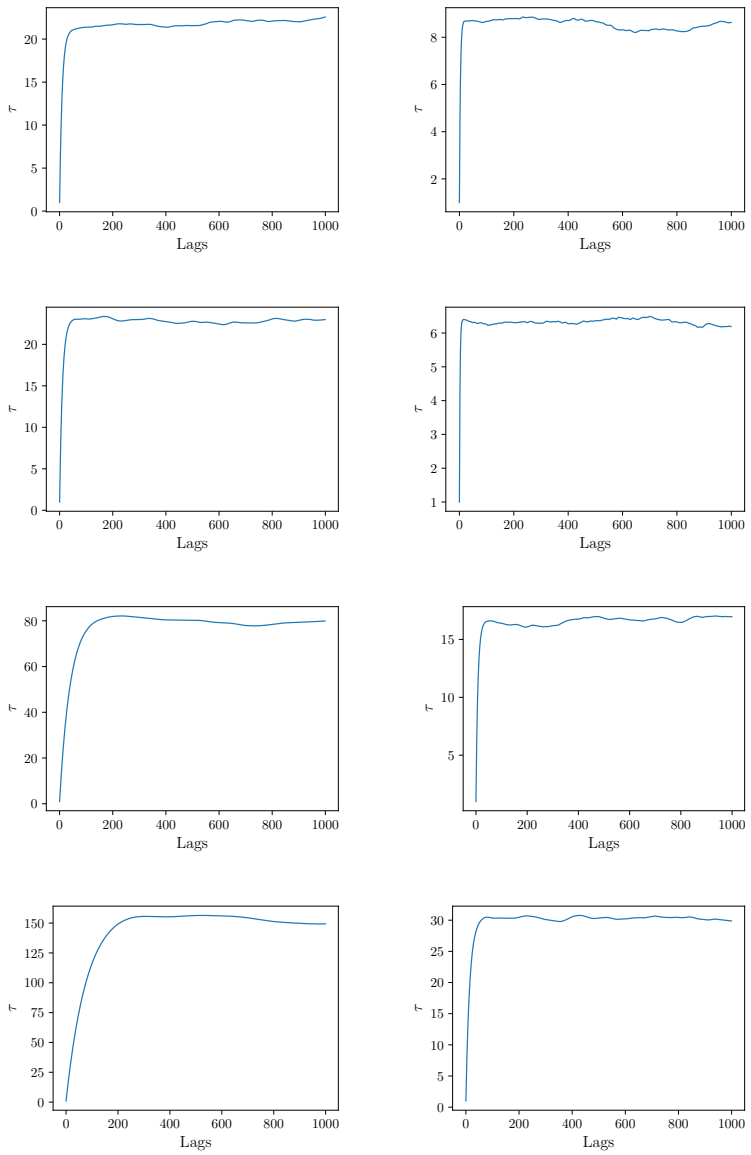



Figure 4.3: Integrated autocorrelation as a function of lag for the sequential approach (left) and MH-MCMC (right) for steps 0.5, 1.0, 5.0, and 10.0 (from top to bottom) scaled by $\eta_{\text{opt}}$.

Figure 4.3 shows the integrated autocorrelation times for the sequential and MH methods for various step-sizes from one of the experiments (i.e. the results from using one of the synthetic datasets $d_{\text{obs}}$). These plots demonstrate that the integrated autocorrelation times have converged reasonably within the first 400 lags. We do not show these results for all experiments and step-sizes here for brevity.

**Efficiency for Each Experiment**

| | 0.5 | 1.0 | 2.0 | 3.0 | 4.0 | 5.0 | 8.0 | 10.0 |
|---|---|---|---|---|---|---|---|---|
| experiment 1 | 0.897 | 0.957 | 1.482 | 1.546 | 1.918 | 2.35 | 3.357 | 4.104 |
| experiment 2 | 0.6 | 0.535 | 0.612 | 0.786 | 0.971 | 1.245 | 1.626 | 2.146 |
| experiment 3 | 0.909 | 0.876 | 1.24 | 1.614 | 2.143 | 2.272 | 3.407 | 4.129 |
| experiment 4 | 0.53 | 0.424 | 0.497 | 0.638 | 0.75 | 0.876 | 1.213 | 1.633 |
| experiment 5 | 0.692 | 0.63 | 0.85 | 1.117 | 1.379 | 1.439 | 2.207 | 3.013 |

Table 4.1: Efficiency from each experiment for step-sizes scaled by $\eta_{\text{opt}}$.

Figure 4.1 shows the efficiency for various step-sizes averaged over the five experiments. In Table 4.1, we show the efficiencies for each experiment. It is clear that the efficiency of the sequential approach is somewhat sensitive to the noise in generating the data. Although the same noise model was used in all experiments, some sets of observational data led to greater efficiency in the sequential approach than others. It is for this reason that we reported averaged values in Figures 4.1 and 4.2. However, in all cases, as step-size increased from the optimal $\eta_{\text{opt}}$, so did the efficiency of the sequential approach relative to MH-MCMC.

### 4.3.4 Discussion of Example Results

The results in the previous section demonstrate an important situation in which the sequential approach can yield increased efficiency over the MH-MCMC method. When step-sizes are taken larger than optimal and when the time-steps of the model provide somewhat redundant information, the sequential approach can be more efficient by spending fewer model solves in areas with low probability since initial time-steps of the model can be used to determine whether or not to reject a move without solving the full model.

Note that when a smaller step-size is used in both the MH and sequential approaches, the MH algorithm tends to be more efficient. In this example, the time series of model output is perfectly redundant since the model is simply the identity map. If we had access to the noise-free data, then a change in parameters would result in likelihood ratios $r_t$ which are either uniformly less than or greater than one. However, the noise in the data obscures

this correlation. When using a small step-size, it is likely that most jumps will result in likelihood ratios $r_t$ which are neither uniformly greater than or uniformly less than one. This is entirely due to the noise in the data. The reason that larger steps yield improved relative performance for the sequential approach is that these steps often explore regions of parameter space for which the change in model outputs for the parameters is larger than the scale of the noise so that the likelihood ratios are more often uniformly greater than or less than one.

Of course, for a general non-Gaussian posterior, an optimal step-size may not be known. The sequential approach could be useful in generating samples from a posterior until a more appropriate step-size can be determined. Furthermore, for high-dimensional and extremely non-Gaussian posteriors it may not be practical to explore the posterior enough to determine an optimal step-size at all. We show in the next chapter how inverse problems using the reservoir model described Chapter 2 can lead to posteriors for which there is no obvious choice of step-size and how the sequential approach can lead to improved performance in this case.

# Chapter 5

# Numerical Results

In this chapter we apply MCMC using the sequential transition kernel described in Chapter 4 to the reservoir problem described in Chapter 2. The reservoir has faults and the transmissibility of the faults are the inversion targets. We assume here that the faults have constant transmissibility although this assumption could be relaxed and one could describe the transmissibilities as fields defined along the faults. First, the forward problem setup will be described. Second, we define the specific inverse problem being solved. Finally, we apply the sequential MCMC algorithm to the inverse problem and compare its performance to that of a MH-MCMC approach.

## 5.1 Forward Problem Setup

We solve the system described in eq. (2.8) for a square 2D reservoir domain. The permeability field is assumed to be uniform and isotropic throughout the reservoir so that it can be described by a single scalar value. In Table 5.1 we summarize the parameters used in the forward model for these results.

| Parameter | Value | Units |
|---|---|---|
| Reservoir length | 500 | $m$ |
| Initial pressure | $1e7$ | $Pa$ |
| Flow rate, (producers) | $1e-4$ | $m^3/s$ |
| Bottom hole pressure, (injector) | $1.2e7$ | $Pa$ |
| Well radius | 0.125[1] | $m$ |
| Reference density, water | 1000 | $kg/m^3$ |
| Reference density, oil | 900 | $kg/m^3$ |
| Reference pressure | $1.01325e5$ | $Pa$ |
| Viscosity, water | $3e-4$ | $Pa\ s$ |
| Viscosity, oil | $6e-4$ | $Pa\ s$ |
| Compressibility, water | $4.35e-10$ | $Pa^{-1}$ |
| Compressibility, oil | $4.35e-10$ | $Pa^{-1}$ |
| Compressibility, rock | $1.45e-10$ | $Pa^{-1}$ |
| Permeability | $1e-13$ | $m^2$ |
| Porosity | 0.3 | |

Table 5.1: Parameters for the reservoir model.

We take the reservoir domain $\Omega$ to be $[0, 500]^2$ in meters with boundary $\partial\Omega$. There are two faults $\Gamma_1$ and $\Gamma_2$. The fault $\Gamma_1$ runs from $[125, 225]$ to $[225, 125]$ while $\Gamma_2$ runs from $[200, 350]$ to $[350, 200]$. The total simulation time is $T \approx 50.53$ months and the time-step discretization is $dt \approx 12.81$ days so that the model runs for 120 time-steps. The relative permeability functions for the water and oil phases are given by $k_{r,w} = S^2$ and $k_{r,o} = (1-S)^2$ respectively. We assume no-flow conditions along the boundary $\partial\Omega$. Finally, the injection well is located in the bottom left corner of the reservoir and is taken to be a pressure well (constant bottom hole pressure). The production wells are located in the remaining three corners and are rate wells. A plot of the geometry of the reservoir domain with faults and wells is shown in Figure 5.1

[1]Exact value of well radius is $0.12512136927444853\,m$. This value was based off the minimum cell diameter of the mesh.

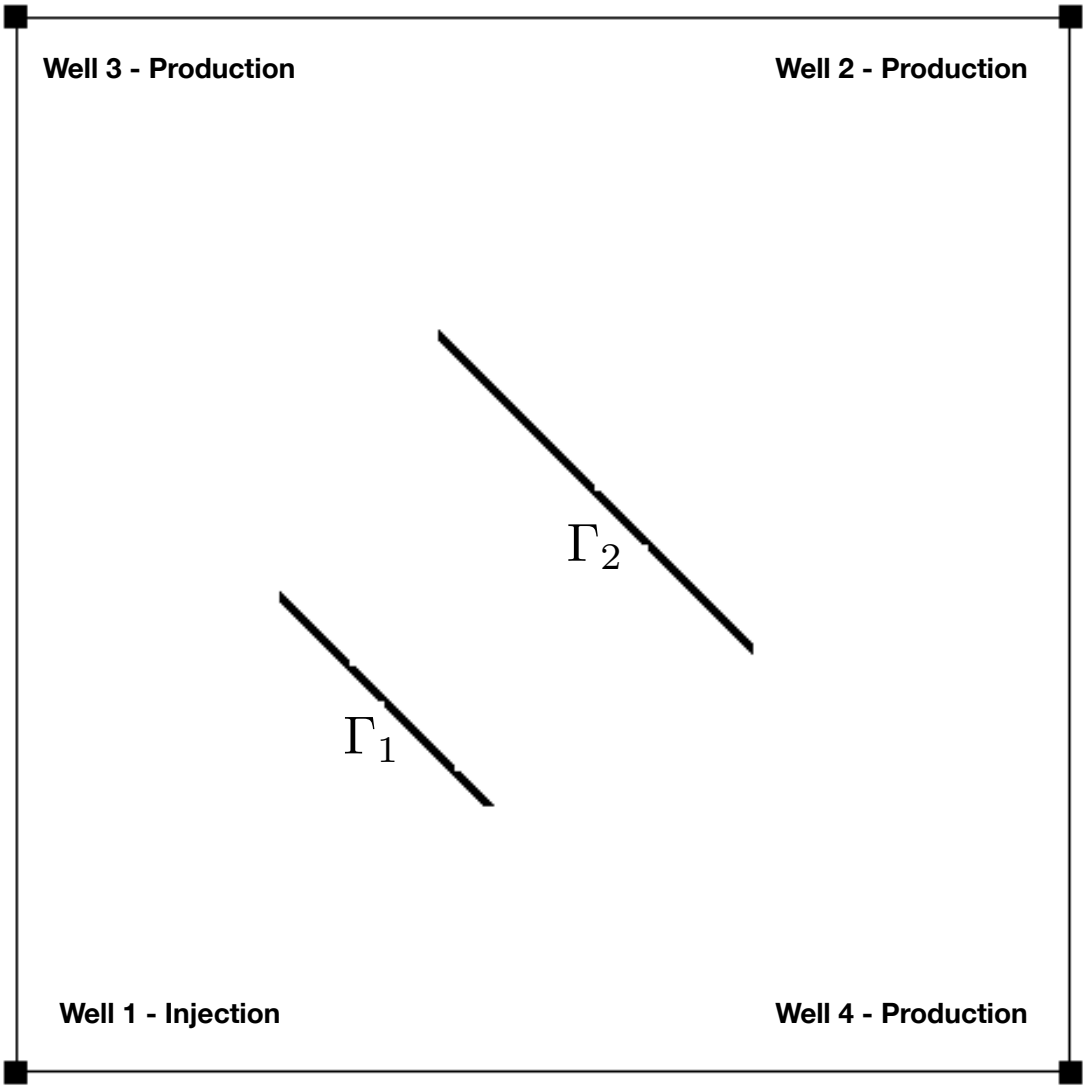



Figure 5.1: Geometry of the reservoir domain. Faults are represented as black lines in the interior of the domain. Wells are black dots at corners. Injection well in bottom-left and producers in the remaining corners.

The mesh for the problem is generated using the finite element mesh generator Gmsh [11]. The mesh is unstructured and conforming to the fault with a mesh size of $25m$. A plot of the mesh is shown in Figure 5.2.

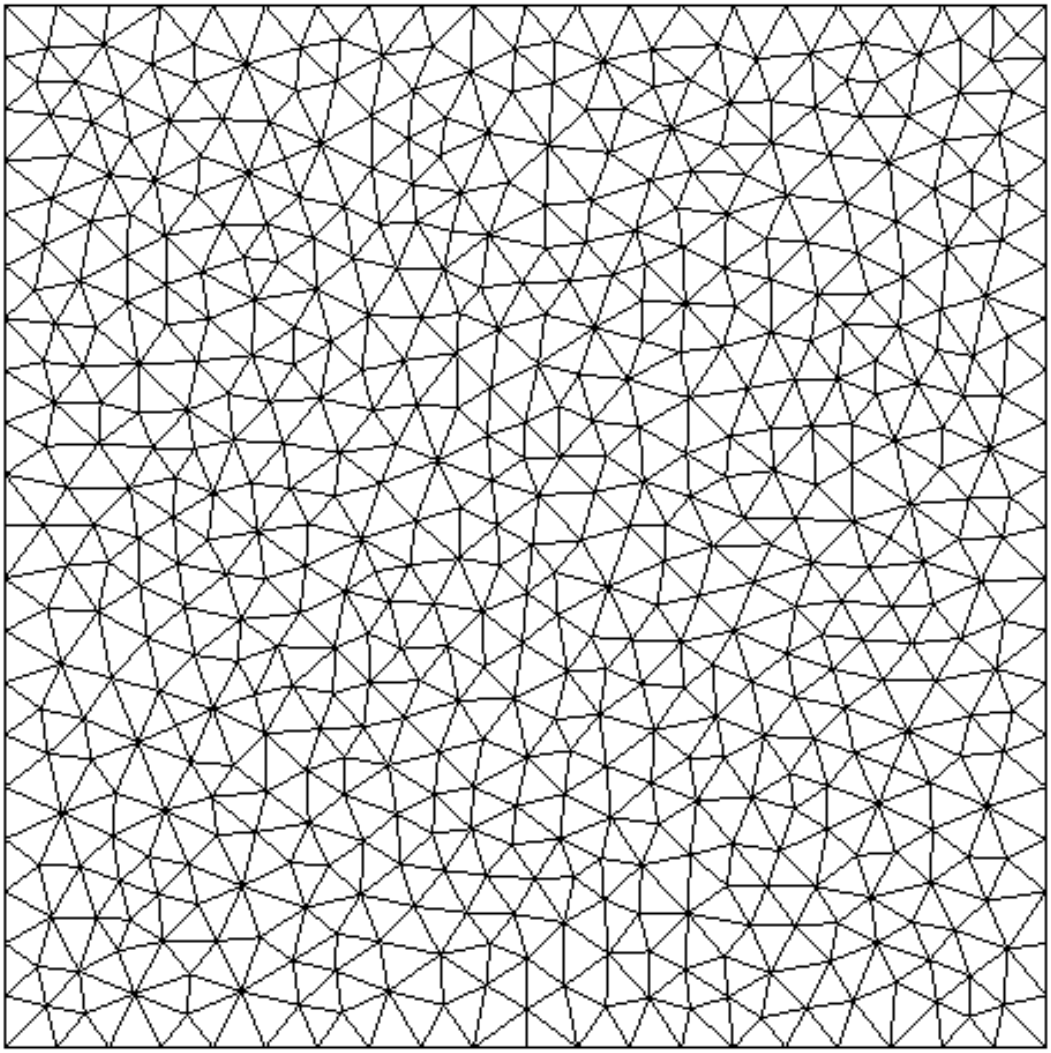
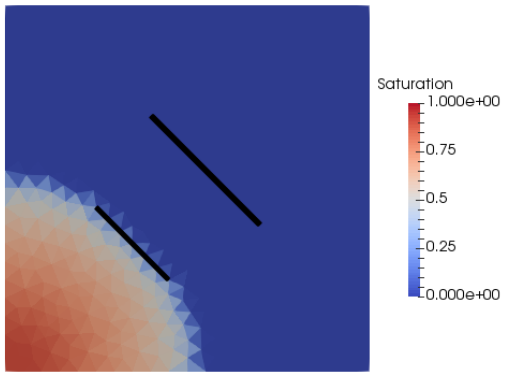
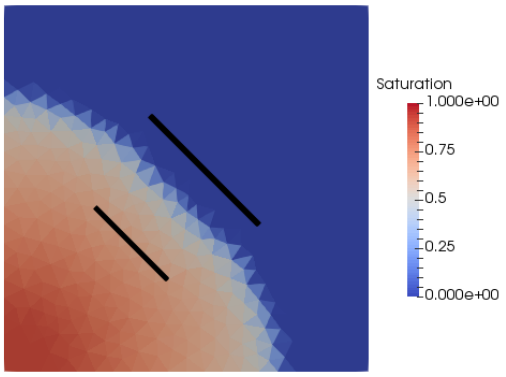


Figure 5.2: Computational mesh with edge length of $25m$ for each triangle. Mesh is unstructured and conforms to the faults.

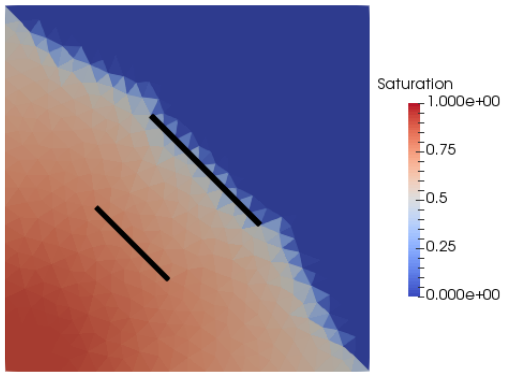
In Figure 5.3 we show the saturation field in the reservoir after 30, 60, 90, and 120 time-steps. This corresponds to roughly 12.5, 25, 37.5, and 50 months from the start of the simulation. The fault transmissibilities in this case are taken to be the true transmissbility used in the inverse problem described in the next section. Figure 5.4 shows the pressure field using the same values of transmissibility after 60 time-steps or approximately 25 months from the start of the simulation.



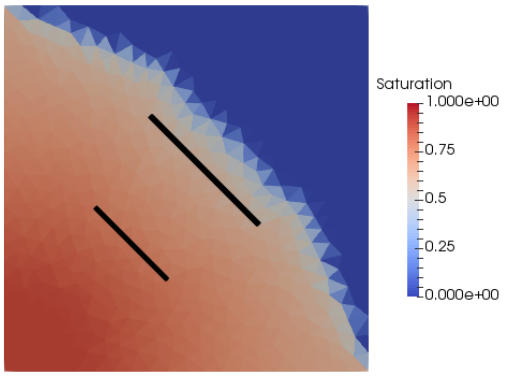

(a) Saturation at 30 time-steps.




(b) Saturation at 60 time-steps.




(c) Saturation at 90 time-steps.




(d) Saturation at 120 time-steps.

Figure 5.3: Water saturations.

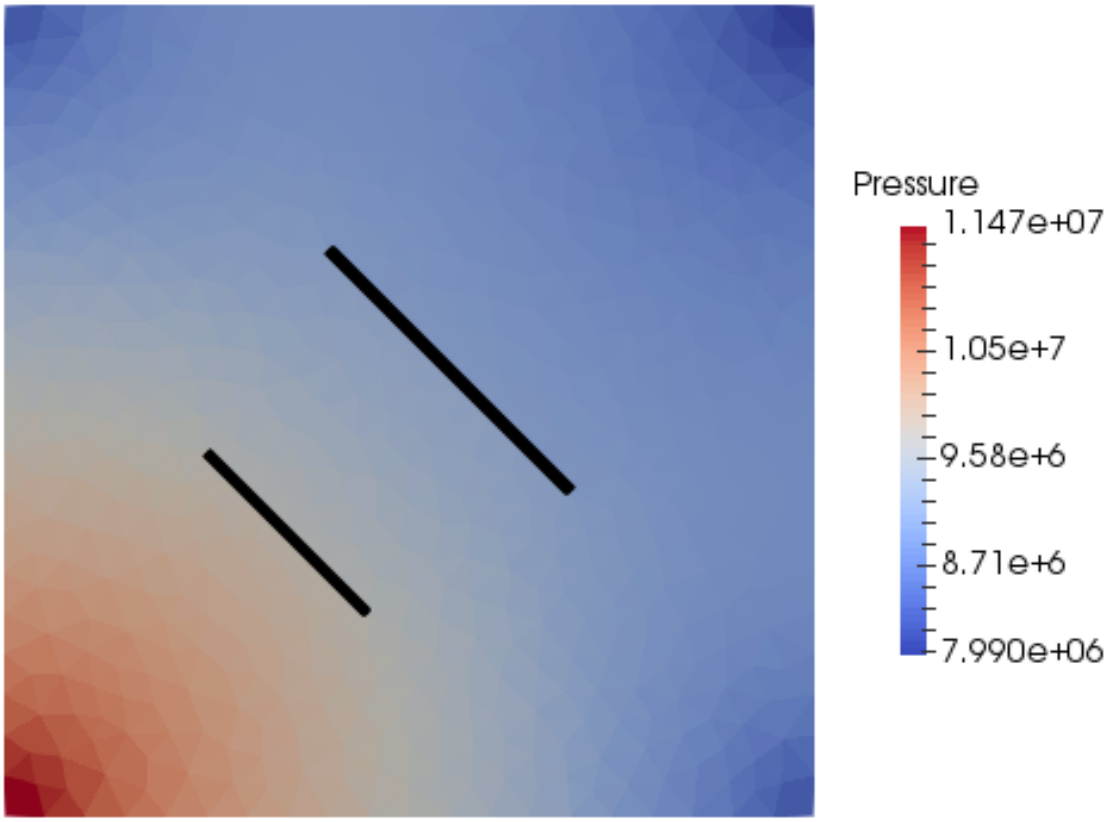



Figure 5.4: Pressure at 60 time-steps.

## 5.2 Inverse Problem Setup

We wish to invert for fault transmissibilities using pressure data at the wells. For convenience, we non-dimensionalize the fault transmissibilities by the quantity $t_{f,0} = K_0/L\mu_0$ where $L$ is the length of one side of the square reservoir domain and $K_0$ and $\mu_0$ are reference permeability and viscosity values respectively. We take the reference permeability equal to the permeability of the reservoir (which is a scalar) and the reference viscosity equal to the wetting phase viscosity $\mu_w$. This gives

$$t_{f,0} = \frac{10^{-13}}{500 \times 3 \times 10^{-4}} = \frac{1}{15} \times 10^{-11}. \tag{5.1}$$

Furthermore, we choose to invert for the log-transmissibility of the faults. Therefore, our inversion targets are $x = \log(t_f/t_{f,0})$ where $t_f$ are the fault transmissibilities. Clearly, there is a one-to-one relationship between $x$ and $t_f$ so that inverting for one is equivalent to inverting for the other.

Similar to the example in Chapter 4.3 we use a true parameter $x_{\text{true}}$ to generate noisy synthetic observed data

$$d_{\text{obs}} = \mathcal{M}(x_{\text{true}}) + \eta \tag{5.2}$$

where $\eta$ is some mean-zero random variable representing the noise in the observation. Here we use

$$x_{\text{true}} = [2.0, 1.5]\,. \tag{5.3}$$

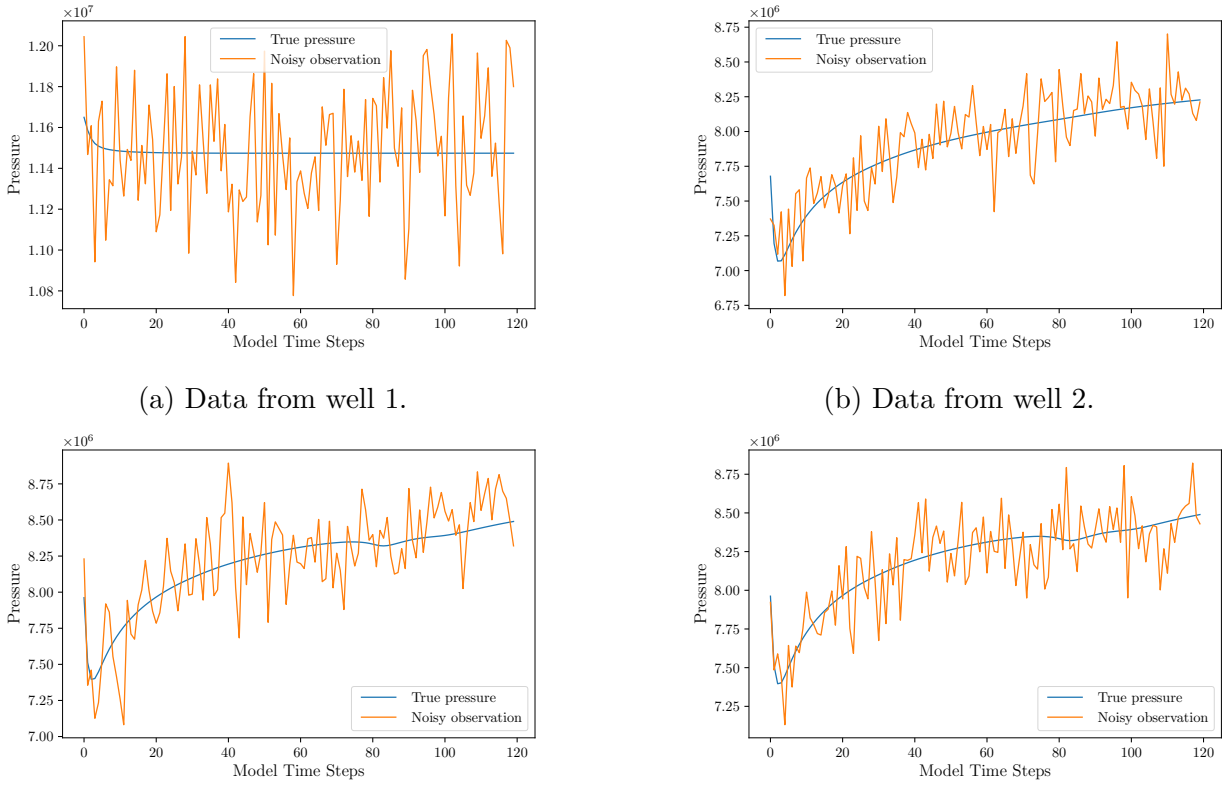
We take $\eta$ to be mean-zero Gaussian noise. The observations are pressure values at the four wells. Let $d^i_{\text{obs}}$ be the time-series of observed pressure data at a well $i$ for $i = 1, 2, 3, 4$ with corresponding noise $\eta^i$. Then we assume that for each well, $\eta^i$, the covariance matrix is given by $\sigma^2 I_T$ where $I_T$ is the $T \times T$ identity matrix and $\sigma$ is the standard deviation of

the noise. We take

$$\sigma = 0.05 * \max(\mathcal{M}^i(x_{\text{true}})), \tag{5.4}$$

where $\mathcal{M}^i$ is the parameter-to-observable map for time-series pressure data at well $i$.

In Figure 5.5 we show the pressure observations generate by $x_{\text{true}}$ for each well along with the noise-corrupted observation $d_{\text{obs}}$.




(a) Data from well 1.

(b) Data from well 2.

(c) Data from well 3.

(d) Data from well 4.

Figure 5.5: Observed pressure data from each well.

We take our prior to be a uniform distribution on $[-8, 12]^2$. The parameter space is only two-dimensional for this problem. We take a grid of parameters over the bounds of the prior and evaluate the posterior. We show the resulting plot of the posterior in Figure 5.6.

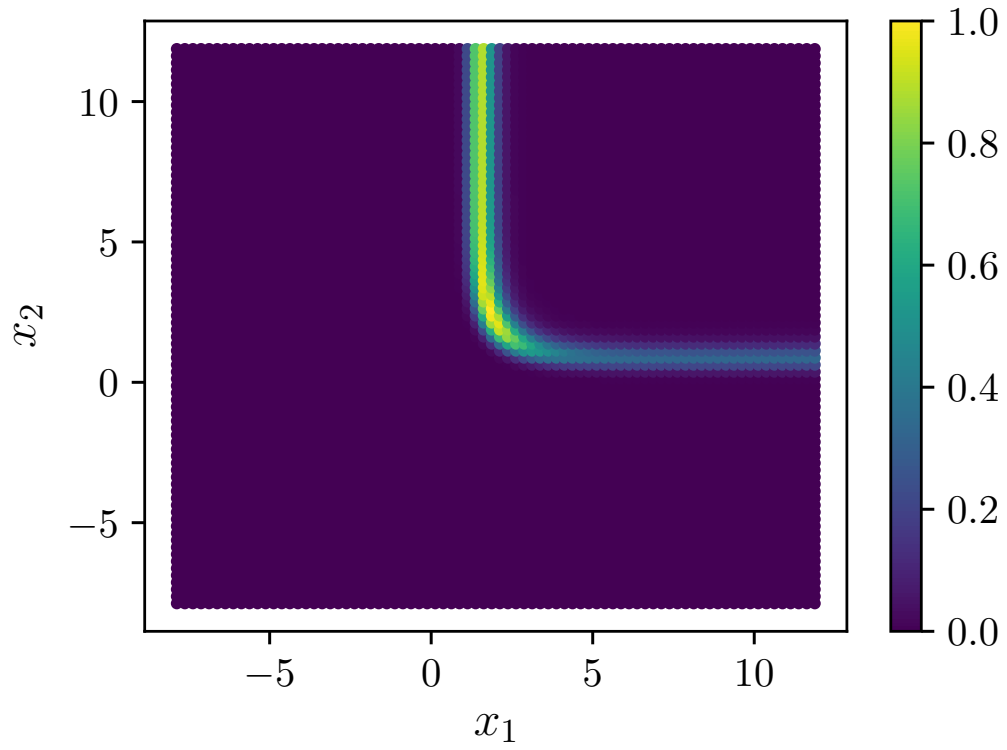



Figure 5.6: Non-normalized posterior for the reservoir problem with uniform prior.

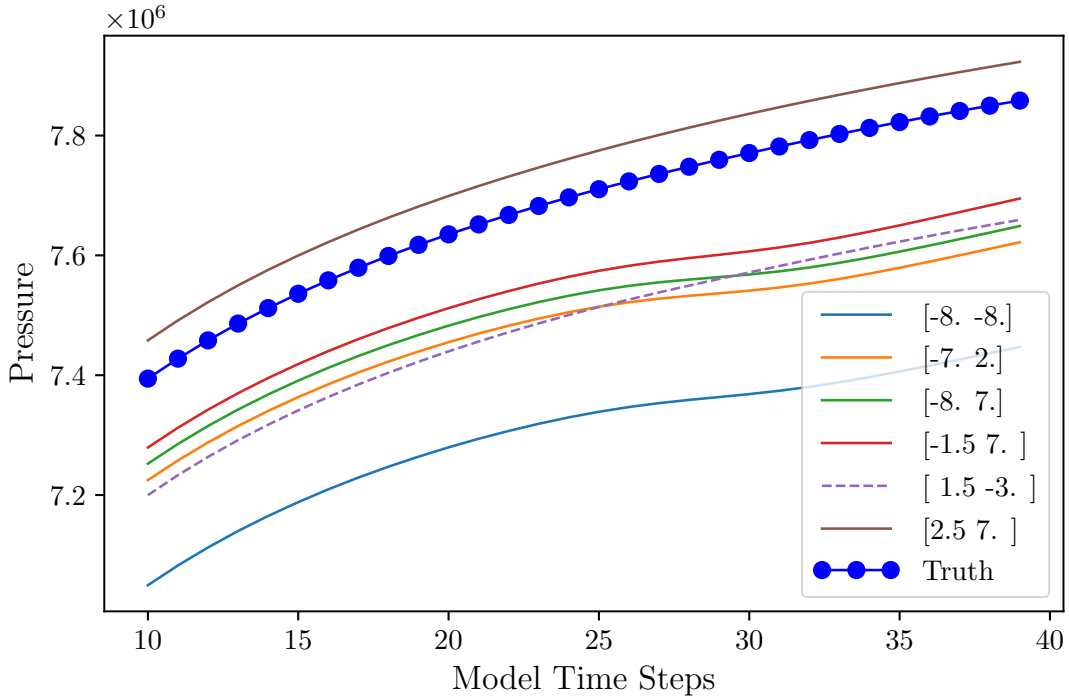
### 5.2.1 Potential for Improved Performance by the Sequential Approach

The sequential approach to MCMC can be more efficient than a standard MH-MCMC approach when two conditions are met. First, the time-series of data should be somewhat redundant, which is a reasonable assumption for many physical models. Second, the step-sizes used should be relatively large. For a Gaussian probability density there is an optimal step-size for MH-MCMC when the proposal distribution is also Gaussian. We demonstrated in Chapter 4.3 that when the posterior distribution is a Gaussian, taking larger than optimal step-sizes led to an increase in the efficiency of the sequential approach relative to MH-MCMC. However, for an arbitrary posterior which may be very non-Gaussian, an optimal choice of step-size may not be obvious. We now describe why the inverse problem solved here may be amenable to the sequential approach to MCMC.

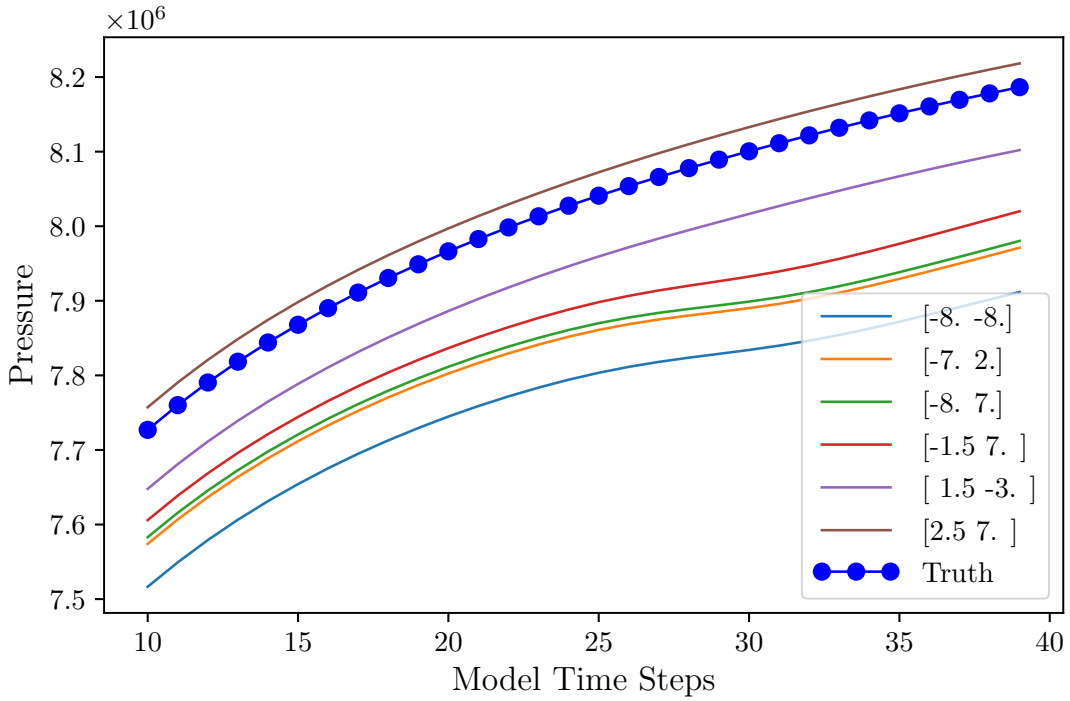
Figure 5.6 shows that the posterior is extremely non-Gaussian and that choosing a single step-size for MCMC may be difficult. For the region of parameter space for which $x_2 > 2.5$ and $0 < x_1 < 5$, a step-size with large variance in the $x_2$ parameter and small variance in $x_1$ may be most efficient for MH-MCMC. However, for the region of parameter space for which $x_1 > 5$ and $0 < x_2 < 2.5$ the reverse is true. A step-size with high variance in the $x_1$ parameter and low variance in the $x_2$ parameter may be preferable. In Chapter 4.3 we showed that using the sequential approach for MCMC can be more efficient than MH-MCMC when larger step-sizes are used in the proposal distribution. The sequential approach allows us to take a large step-size and rely on initial time-steps of the model to reduce model solves when the proposal moves the chain into very low-probability regions of the posterior. We will show that the sequential approach with a relatively large step size outperforms standard MH-MCMC for a variety of step sizes.

Now, the assumption behind the sequential approach is that model output from initial time-steps often contains some information about future time-steps. If this is not the case,

we should not expect the sequential approach to yield improved performance. To determine whether or not one expects the reservoir problem to be a good candidate for the sequential approach, one could examine some of the pressure curves generated by various parameters to see if this is a reasonable assumption. In Figure 5.7 we show pressure curves at wells 1 and 2 generated from various parameters. The parameter values which led to each of the pressure curves are shown in the legend. The pressure curve generated by the true parameter $x_{\text{true}} = [2.0, 1.5]$ is also shown. For ease of presentation, these curves are only shown for time-steps 10 to 39 which corresponds to a time horizon of about 384.3 days.




(a) Model outputs from well 2.




(b) Data from well 3.

Figure 5.7: Model outputs from each well.

What Figure 5.7 shows is that, over the time window, many of the parameters shown lead to pressure curves which do not cross each other so that initial time-steps can be used to decide whether or not a jump leads to a large decrease in posterior probability. To be more precise, consider fig. 5.7b. If one had access to noiseless data, i.e. the blue dotted line in the figure, one would only need to evaluate one time-step of the model for each of these parameters to decide which one produces model output which is closest to the (noise-free) data. In fig. 5.7a we see that of the pressure curves shown, only the one corresponding to parameter $x = [1.5, -3]$ crosses some of the others. This pressure curve is an example of a situation in which the entire time-series of data may be necessary to decide whether or not the given parameter leads to model output closer to or farther away from the data (in a least-squares sense). In Figure 5.7 we have only shown pressure curves from a few parameters at two of the four wells for a small window of time. This is for ease of presentation as displaying more pressure curves over a longer time-window would be too crowded. The purpose of Figure 5.7 is to provide some intuition for situations in which one might expect the sequential approach to perform well relative to MH-MCMC and why the inverse problem considered here may be one such situation.

In summary, we expect the inverse problem considered here to be one for which the sequential approach to MCMC is a better choice than MH-MCMC for two reasons. First, the shape of the posterior means that choosing a single global step-size for which MH-MCMC performs well in all regions of the posterior may be difficult. Second, the time-series data seems to have some redundancy which we can exploit with a sequential approach. Using the sequential MCMC algorithm allows one to take larger step-sizes and rely on initial time-steps to reject proposed jumps which occur in low-probability regions of the posterior.

## 5.3 Results

We run the MH-MCMC approach with isotropic step-sizes of 3.0, 5.0, 7.5, and 10.0 and we run the sequential approach to MCMC with an isotropic step-size of 10. We only choose one step-size for the sequential approach because we expect large step-sizes to work best for the method. Further, even if a smaller step-size would lead to increased performance of the method, we expect this improvement to be slight. We initialize all chains at $[0, 0]$ and run each of the MH chains for 25200 iterations and the sequential chain for 250200 iterations. For each chain, we discard the first 200 samples to allow for a burn-in period. This means that we effectively have 25000 samples for each of the MH chains and 250000 samples for the sequential method. The prior is uniform, so for both methods, when a jump is proposed outside the bounds of the prior, we reject without solving the model.

We compare the efficiency of each method as in Chapter 4.3 by computing

$$\epsilon = \frac{n_{solves}^{MH}\,\tau_{MH}}{n_{solves}^{seq}\,\tau_{seq}}, \tag{5.5}$$

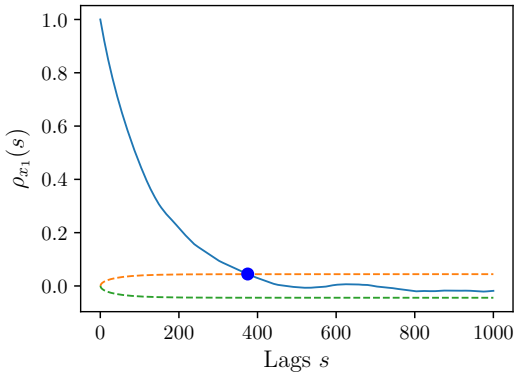
where $n_{solves}^{MH}$ and $n_{solves}^{seq}$ are the average number of solves per sample for MH and sequential MCMC respectively and $\tau_{MH}$ and $\tau_{seq}$ are the integrated autocorrelation times for MH and sequential MCMC respectively. To see why this quantity is used for comparing the methods see eqs. (4.14) and (4.15). In this problem, because the prior is uniform, for both methods when a jump is proposed outside the bounds of the prior it is rejected with zero model solves. This means that $n_{solves}^{MH}$ will not necessarily be equal to the number of time-steps of the model as it was in the example in Chapter 4 but will instead be some number between 0 and 120. For this problem, we compute $\tau_{x_1}$ and $\tau_{x_2}$, the integrated autocorrelation times for the mean of each of the two parameters.

In practice, it is somewhat difficult to make an accurate comparison of the effectiveness

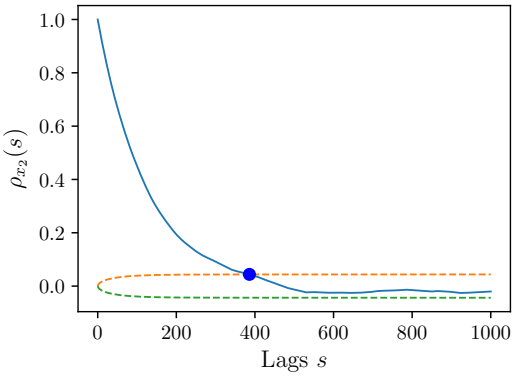
of the two methods due to the noise in the autocorrelation function $\rho(s)$ for a given lag $s$. Resolving the noise for a large number of lags requires generating many samples which in turn requires solving the model many times which can be prohibitive. In order to account for this, we carefully choose how many lags to include in the summation from eq. (4.12) when computing the autocorrelation times $\tau_{x_1}$ and $\tau_{x_2}$ for each chain. To determine an appropriate number of lags for each integrated autocorrelation time, we compute 95% confidence bounds for the autocorrelation function $\rho(s)$ for each lag $s < 1000$. We then find the smallest number of lags $s_0$ for which the value of $\rho(s_0 + 1)$ is smaller than half of the confidence interval. For each integrated autocorrelation time $\tau$ we use $s_0$ lags in the summation so that the noise in each term is not too large relative to the value of $\rho(s)$. That is, for each chain in each parameter, we compute

$$\tau = 1 + 2\sum_{s=1}^{s_0} \rho(s), \tag{5.6}$$

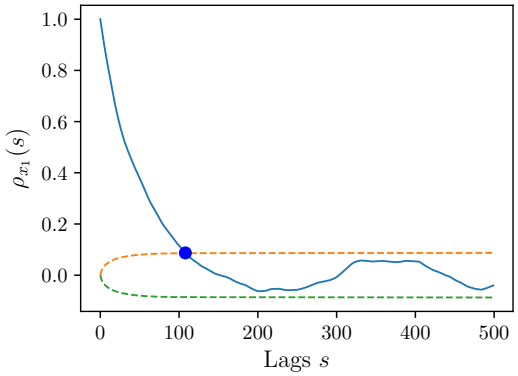
Figure 5.8 shows the autocorrelation functions for each chain in both parameters $x_1$ and $x_2$. The confidence intervals are shown as dotted lines. Once the autocorrelation function falls below this threshold we can be reasonably sure that the noise is larger than the signal and we do not use any more lags in the computation of the integrated autocorrelation times $\tau$ in the results. The largest number of lags before which this happens, $s_0$, is shown in each plot in Figure 5.8 as a blue dot. As in Chapter 4.3 we use the autocorrelation function in the *statsmodels* python package [15].



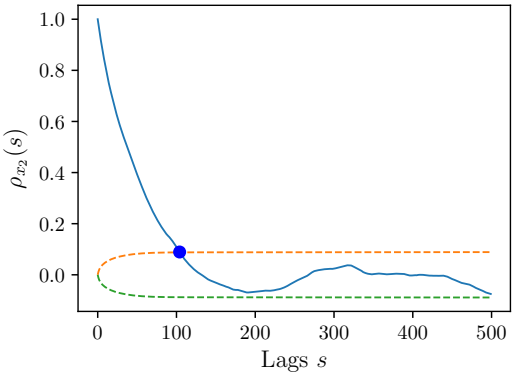

(a) Autocorrelation function for $x_1$ from sequential chain.



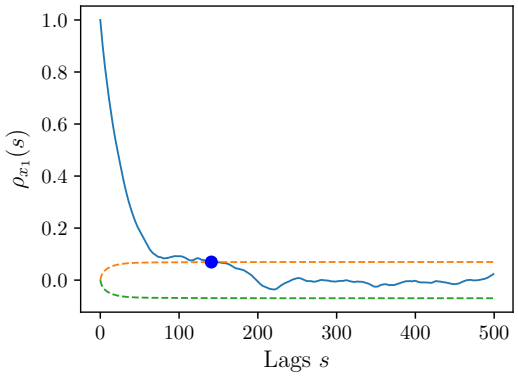

(b) Autocorrelation function for $x_2$ from sequential chain.



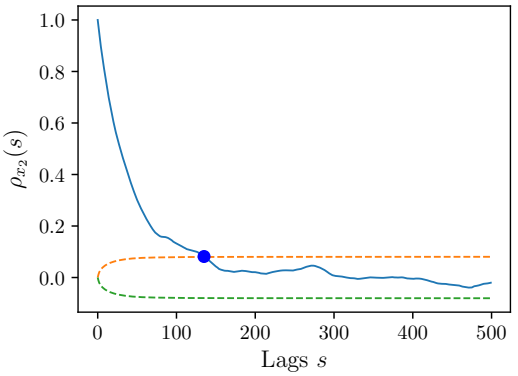

(c) Autocorrelation function for $x_1$ from MH chain with step-size 3.0.




(d) Autocorrelation function for $x_2$ from MH chain with step-size 3.0.

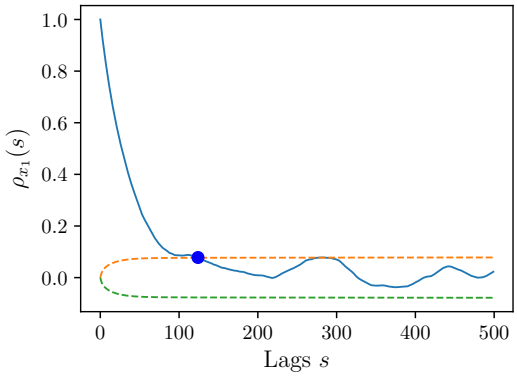



(e) Autocorrelation function for $x_1$ from MH chain with step-size 5.0.

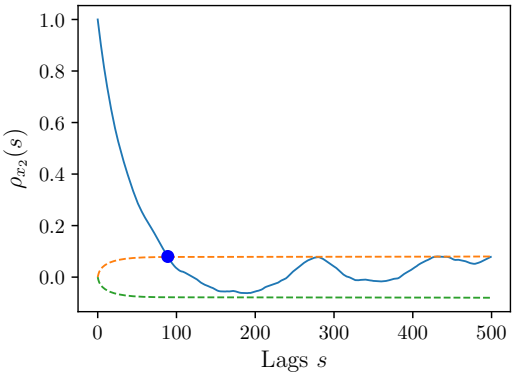



(f) Autocorrelation function for $x_2$ from MH chain with step-size 5.0.

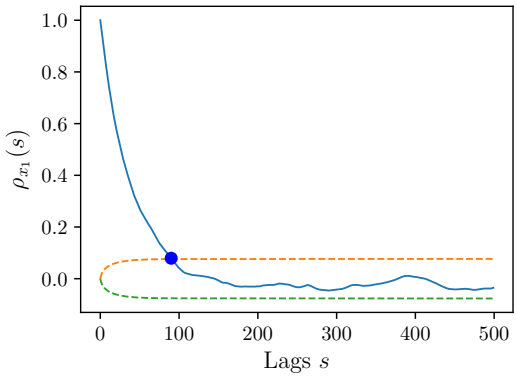
Figure 5.8



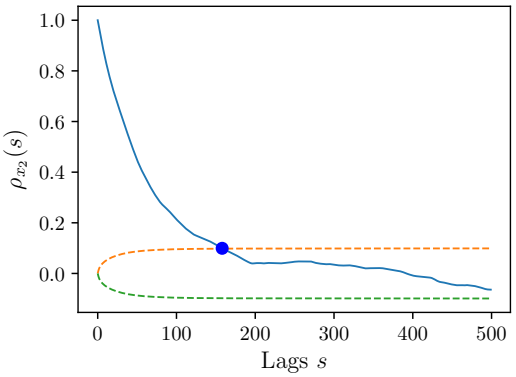

(g) Autocorrelation function for $x_1$ from MH chain with step-size 7.5.




(h) Autocorrelation function for $x_2$ from MH chain with step-size 7.5.




(i) Autocorrelation function for $x_1$ from MH chain with step-size 10.0.




(j) Autocorrelation function for $x_2$ from MH chain with step-size 10.0.

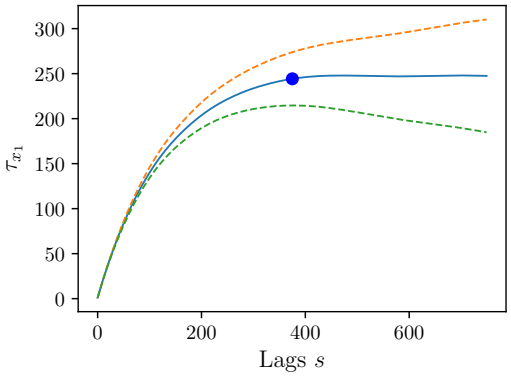
Figure 5.8: Autocorrelation functions for each chain in the experiment for both parameters $x_1$ and $x_2$. The dotted lines are 95% confidence bounds and the blue dots are $s_0$.

| | Sequential | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|---|
| $x_1$ | 375 | 108 | 141 | 124 | 90 |
| $x_2$ | 386 | 104 | 135 | 89 | 158 |

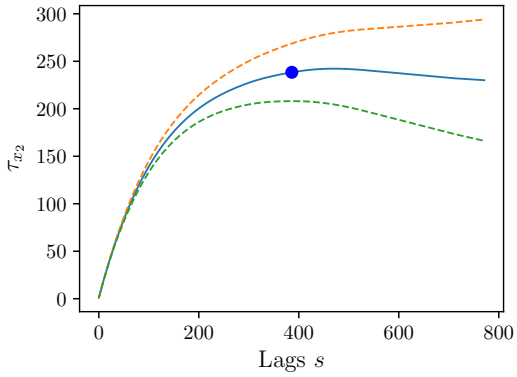
Table 5.2: Maximum number of lags $s_0$ used in computing the integrated autocorrelation times for each chain in parameters $x_1$ and $x_2$.

Table 5.2 shows $s_0$ for each chain in each variable. This is the maximum number of lags we use when computing the integrated autocorrelation times $\tau$ for each chain.

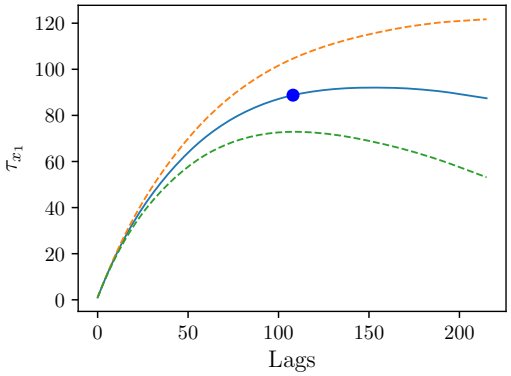
In Figure 5.9 we show the integrated autocorrelation times for each of the chains in both parameters for $2s_0$ lags. The dotted lines in Figure 5.9 are the integrated autocorrelation times when the 95% confidence bounds in the autocorrelation function are used in eq. (5.6). These show that the integrated autocorrelation times have converged reasonably well in $s_0$ lags.



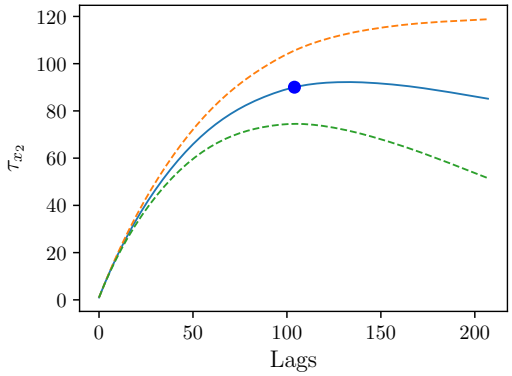

(a) Integrated autocorrelation time for $x_1$ from sequential chain.



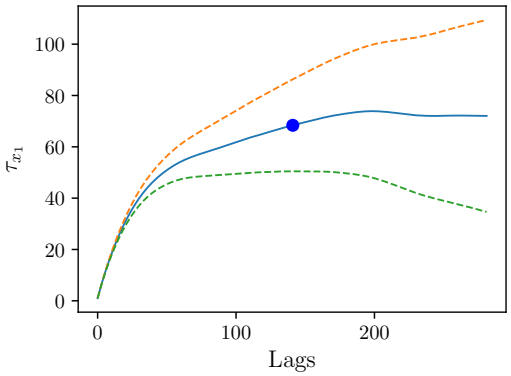

(b) Integrated autocorrelation time for $x_2$ from sequential chain.



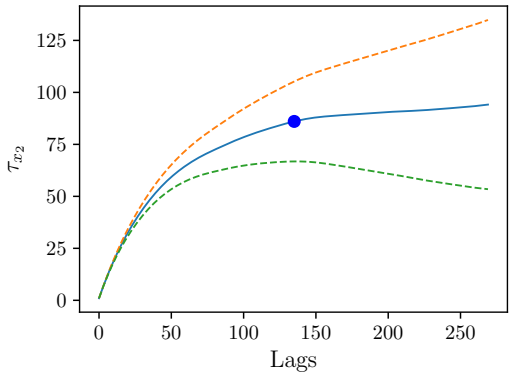

(c) Integrated autocorrelation time for $x_1$ from MH chain with step-size 3.0.




(d) Integrated autocorrelation time for $x_2$ from MH chain with step-size 3.0.

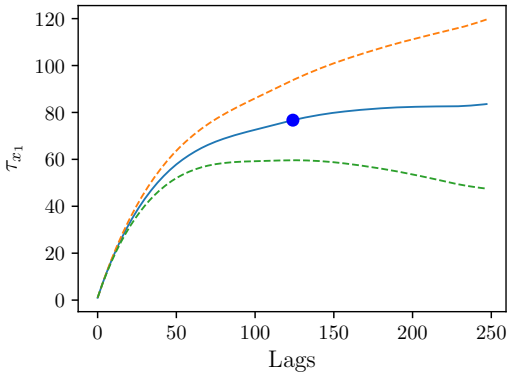



(e) Integrated autocorrelation time for $x_1$ from MH chain with step-size 5.0.

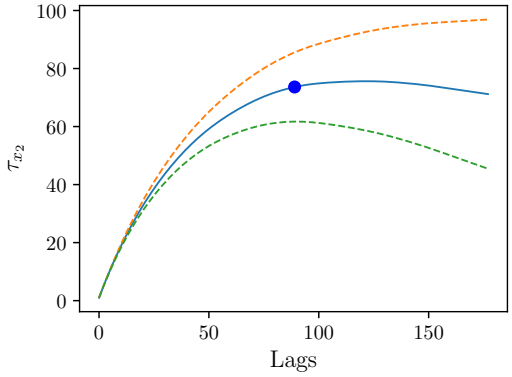



(f) Integrated autocorrelation time for $x_2$ from MH chain with step-size 5.0.

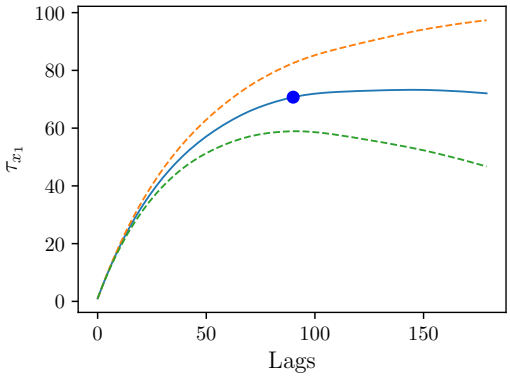
Figure 5.9



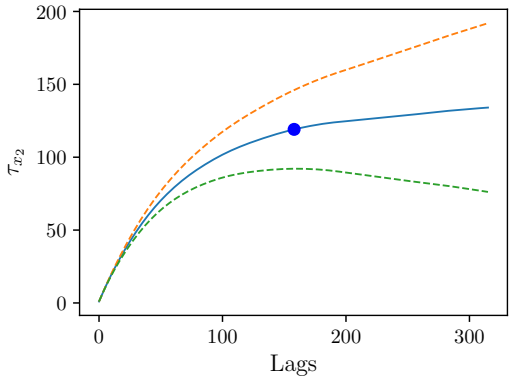

(g) Integrated autocorrelation time for $x_1$ from MH chain with step-size 7.5.




(h) Integrated autocorrelation time for $x_2$ from MH chain with step-size 7.5.




(i) Integrated autocorrelation time for $x_1$ from MH chain with step-size 10.0.




(j) Integrated autocorrelation time for $x_2$ from MH chain with step-size 10.0.

Figure 5.9: Integrated autocorrelation times for each chain in the experiment for both parameters $x_1$ and $x_2$. The dotted lines are computed using 95% confidence bounds in the autocorrelation function $\rho$ and the blue dots are $s_0$.

| | Sequential | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|---|
| Worst Case | 1,334.62 | 11,056.85 | 8,122.5 | 6,484.94 | 4,118.11 |
| Expected | 1,190.3 | 9,378.6 | 6,435.28 | 5,303.43 | 3,529.18 |
| Best Case | 1,045.98 | 7,700.35 | 4,748.06 | 4,121.91 | 2,940.25 |

Table 5.3: $\tau_{x_1}\, n_{solves}$ for each of the methods used. Worst and best cases are computed using the 95% confidence bounds in the autocorrelation function.

| | Sequential | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|---|
| Worst Case | 1,309.49 | 11,151.82 | 9,905.64 | 5,915.18 | 7,289.35 |
| Expected | 1,161.72 | 9,511.99 | 8,096.76 | 5,090.9 | 5,942.7 |
| Best Case | 1,013.95 | 7,872.16 | 6,287.87 | 4,266.62 | 4,596.05 |

Table 5.4: $\tau_{x_2}\, n_{solves}$ for each of the methods used. Worst and best cases are computed using the 95% confidence bounds in the autocorrelation function.

| | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|
| Best Case | 10.57 | 7.77 | 6.2 | 3.94 |
| Expected | 7.88 | 5.41 | 4.46 | 2.96 |
| Worst Case | 5.77 | 3.56 | 3.09 | 2.2 |

Table 5.5: Relative efficiency $\epsilon$ of the sequential approach compared to MH with various step-sizes in estimating parameter $x_1$.

| | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|
| Best Case | 11 | 9.77 | 5.83 | 7.19 |
| Expected | 8.19 | 6.97 | 4.38 | 5.12 |
| Worst Case | 6.01 | 4.8 | 3.26 | 3.51 |

Table 5.6: Relative efficiency $\epsilon$ of the sequential approach compared to MH with various step-sizes in estimating parameter $x_2$.

In Tables 5.3 and 5.4 we show the quantities $\tau_{x_1} n_{solves}$ and $\tau_{x_1} n_{solves}$ respectively for each method using the lags in Table 5.2. The best and worst case values are computed using the lower and upper 95% confidence bounds in the autocorrelation function while the expected values are computed using the expected values of $\rho(s)$.

Tables 5.5 and 5.6 shows the corresponding relative efficiencies of the sequential MCMC approach with a step-size of 10.0 compared to MH with step-sizes 3.0, 5.0, 7.5, and 10.0.

These show that, the MH chain with a step-size of 10.0 seems to perform best among all the MH chains when estimating $x_1$ while the MH chain with a step-size of 7.5 is slightly more efficient than the other MH chains for estimating $x_2$. The sequential approach is more efficient than any of the MH chains for estimating both parameters $x_1$ and $x_2$. Even in the worst case, it is roughly 220% more efficient than the MH approach when estimating the parameter $x_1$ and 320% more efficient when estimating $x_2$. Furthermore, the worst case efficiencies are extremely pessimistic. We expect that the actual integrated autocorrelation times are well within the confidence bounds computed.

| | Sequential | MH 3.0 | MH 5.0 | MH 7.5 | MH 10.0 |
|---|---|---|---|---|---|
| Acceptance | 0.012 | 0.118 | 0.066 | 0.037 | 0.023 |
| Acceptance per Solve | 0.283 | 0.134 | 0.084 | 0.064 | 0.056 |
| Average Jump | 29.341 | 7.877 | 18.18 | 29.189 | 32.956 |

Table 5.7: Table shows the average acceptance rate, acceptance rate per model solve, and mean squared jump distance per accepted sample for each method.

Table 5.7 gives the acceptance rates for each method. It also shows the acceptance rate per full model solve. We emphasize that, up to this point, we have considered a model solve to be computing one time-step of the model. In this case, we report the number of accepted samples per full model solve (i.e. evaluating 120 time-steps of the reservoir model). This is a natural metric considering any time we accept a proposed jump with the sequential approach we must evaluate the full model. We note that, while the acceptance rates for each method are extremely low, the acceptance rate per full model solve for the sequential approach is roughly 28.3%. To give some context to this number, for a two-dimensional Gaussian target distribution and a MH MCMC algorithm with Gaussian proposal, optimal acceptance rates are roughly 35% [16]. The sequential approach has an acceptance rate per full model solve fairly close to this value while the MH approaches even with relatively small step-sizes are much worse. Finally, we consider the mean squared jump distance per accepted sample. It is common to compute the mean squared jump distance for a chain as it can be an indicator of how well the chain is mixing. In this case, only considering the mean squared jump distance of accepted samples is more natural because iterations of each method require different numbers of model solves on average. Since we are interested in how well the chain is mixing per model solve, considering acceptance rates per full model solve together with the mean squared jump distance per accepted sample provides some insight into how efficiently the method is performing.

Figure 5.10 shows the distribution of samples obtained from the sequential approach along with the distribution of samples from the MH-MCMC approach with a step-size of

10.0.

We use an evenly spaced grid of 6400 points to estimate the mean and variance of the posterior denoted $\mu_{\text{true}}$ and $\sigma^2_{\text{true}}$ respectively. We show in Figure 5.11 how the estimates of the mean of both parameters obtained from the samples of the sequential chain converge to the correct values as the number of samples increases.

The mean of the samples from the full sequential chain in each parameter is $\mu_{\text{seq}} \approx [3.1656, 4.9176]$ while the grid mean for each parameter is $\mu_{\text{true}} \approx [3.1147, 4.9552]$ so that the difference between them is

$$|\mu_{\text{seq}} - \mu_{\text{true}}| \approx [0.0509, 0.0376]\,. \tag{5.7}$$

The integrated autocorrelation times for each parameter in the sequential chain are $\tau_{x_1} \approx 244.26$ and $\tau_{x_2} \approx 238.40$ so that the effective sample size ($n_{\text{eff}}$) for the sequential chain in each parameter is

$$n_{\text{eff}} \approx [1023, 1049]\,. \tag{5.8}$$

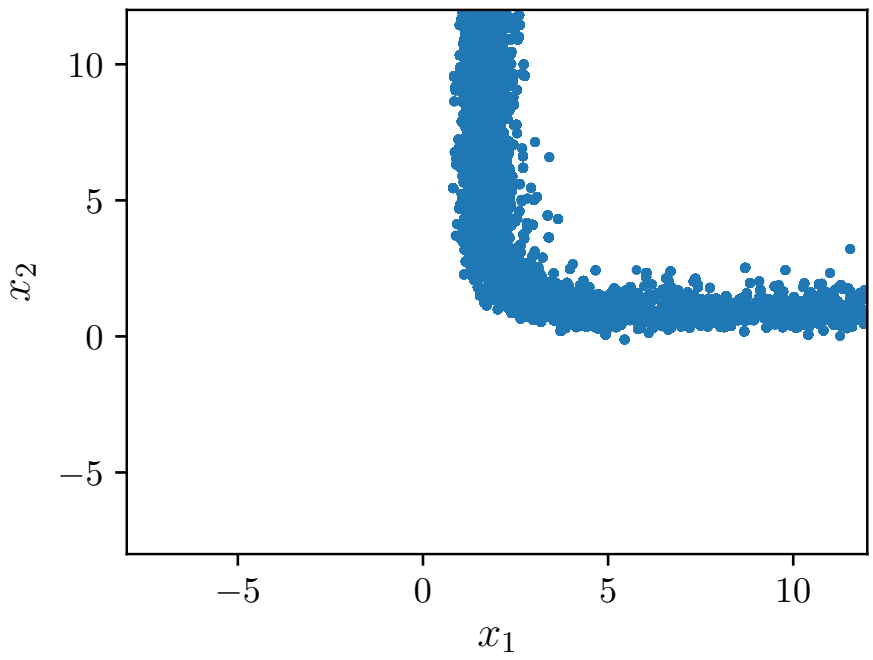
The variance of each parameter computed using the grid of samples is

$$\sigma^2_{\text{true}} \approx [7.3742, 12.9291]\,. \tag{5.9}$$

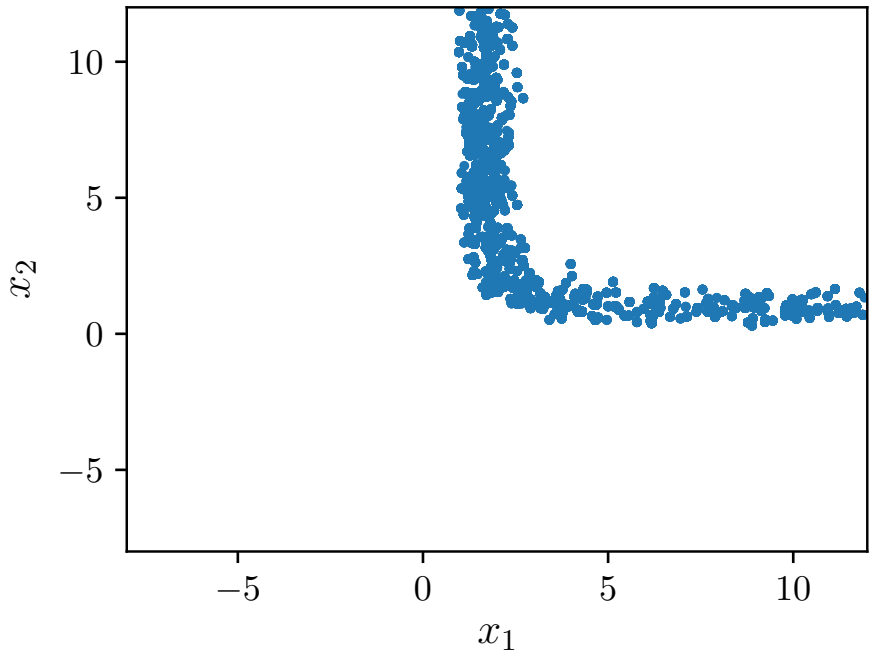
Now, using the effective sample size in eq. (5.8) along with the grid variance in eq. (5.9), we can estimate the standard deviation of the sample mean error of the sequential chain [12] as

$$\sqrt{\frac{\sigma^2_{\text{true}}}{n_{\text{eff}}}} \approx [0.0849, 0.1110]\,. \tag{5.10}$$

From eqs. (5.7) and (5.10), we see that the difference between the sample mean of the sequential chain and the grid mean is within a standard deviation of the sample mean error computed using the integrated autocorrelation times for the sequential chain and from the




(a) Sequential chain.




(b) MH chain.

Figure 5.10: MCMC chains for the sequential and MH approaches with step-sizes of 10.0 each.

grid means and variances. This indicates that the sample means of the sequential chain are converging to the correct means for parameters $x_1$ and $x_2$ as expected.

Finally, we note that the variance of the samples from the sequential chain for each parameter is $\sigma^2_{\text{seq}} \approx [7.7297, 12.8770]$. The difference between the sample variance from the sequential chain and the variance from the grid of samples is

$$|\sigma^2_{\text{true}} - \sigma^2_{\text{seq}}| \approx [0.3555, 0.0521]\,. \tag{5.11}$$

This shows that the sample variance of the sequential chain agrees well with the grid variance.

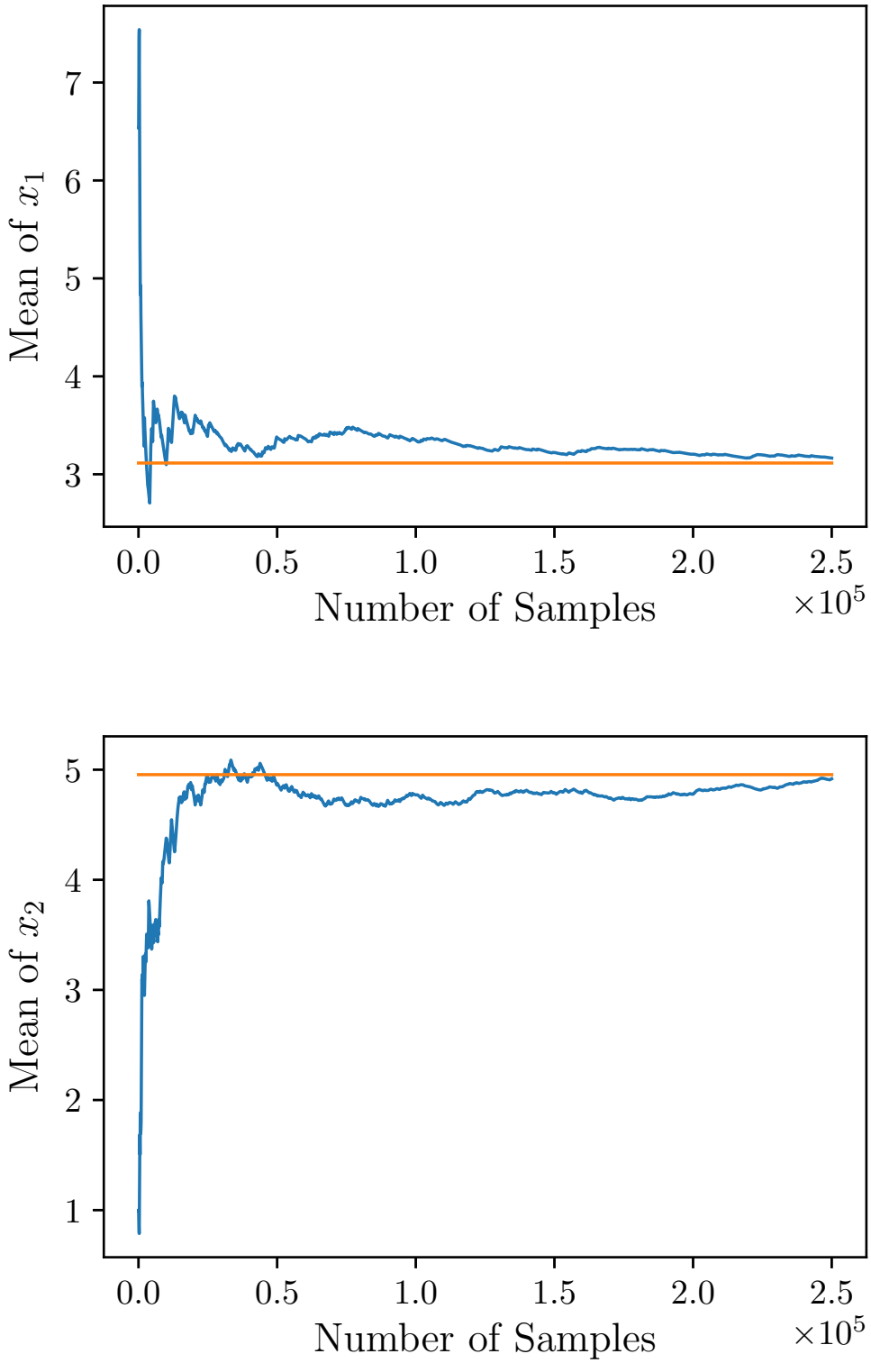



Figure 5.11: Convergence of sequential sample means to grid means. The sequential chain sample means are the solid blue lines. Grid means are shown as flat solid orange lines.

## 5.4 Discussion

We have shown that the sequential approach with a large step-size is more efficient for this problem than MH-MCMC for a variety of step-sizes. The reasons for this improved performance are the redundancy of time-series data in this reservoir model and the shape of the posterior which makes choosing a single step-size difficult. The fact that model output from initial time-steps is often correlated with future time-steps allows one to reject steps which lead to extremely low-probability regions of parameter space with relatively few model solves. The posterior having two distinct, elongated tails necessitates this larger step-size.

Crucial to this comparison is the limitation that both methods use a single step-size for the entire parameter space. In principle, one could choose multiple step-sizes for different regions of parameter space to improve performance. In this case, one would expect the sequential approach to be less efficient than MH-MCMC as the choice of such a proposal would depend on the shape of the posterior which implicitly contains information about the entire time-series of data. In practice choosing such a step-size would require significant exploration of the parameter space and would be extremely impractical for even moderately expensive models in higher dimensions. The utility of the sequential approach is that it allows one to choose a single step-size which is very large and use information from initial time-steps to reduce model-solves in low-probability regions which makes it extremely practical for problems such as the one considered here.

We acknowledge that choosing a step-size which is not isotropic could lead to increased performance in MH-MCMC. The posterior has some symmetry about the diagonal $x_1 = x_2$ but more probability mass in the region where $x_2 > 2.5$. To account for this it may be more appropriate to choose a step-size which is tuned for that region of parameter space more than the other regions. Finding such a step-size would have required running many more MH-MCMC chains to determine one with an optimal step-size in both directions. This was

beyond the scope of the current work but may be interesting for future research. We expect that improved performance from this anisotropic step-size for MH-MCMC would be fairly modest as the posterior is somewhat symmetric as noted earlier. Due to the magnitude of the relative efficiency of the sequential approach to the MH algorithm for all step-sizes we believe that the sequential approach would still outperform MH-MCMC even with a well-chosen anisotropic step-size. Regardless, as for choosing multiple step-sizes for different regions of parameter space, choosing such an anisotropic step-size could be difficult and would require knowledge of the shape of the posterior which, in practice, could only be obtained by exploring parameter space using MCMC or some other approach. So at worst, the sequential approach seems to lead to improved performance for isotropic step-sizes and could be used for initial exploration of parameter space until one obtains more knowledge about the shape of the posterior and can choose a step-size accordingly.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

Accurate numerical simulation of reservoirs can help optimize the recovery of hydrocarbons. In order to simulate reservoir behavior, it is often the case that model parameters which cannot be directly measured must be estimated from a combination of prior knowledge and observed reservoir behavior.

In this work, we have presented an MCMC algorithm which can efficiently generate samples from a Bayesian posterior when the likelihood involves time-series data. The connection between the efficiency of this sequential approach relative to a standard Metropolis-Hastings MCMC algorithm and the step-size used in the proposal distribution was demonstrated through a simple problem in which the optimal step-size was known. This sequential MCMC algorithm was then applied to a Bayesian inverse problem in which the forward model was a two-phase reservoir model. We demonstrated that the sequential MCMC approach presented here was more efficient than Metropolis-Hastings for a variety of step-sizes when applied to this reservoir inverse problem.

We acknowledge that the sequential approach has some limitations. It relies on the assumption that time-series data has some redundancy. Although this assumption need not hold exactly, for models in which data at different time-steps informs distinct directions in

parameter space, we expect the sequential approach presented here to perform poorly relative to a standard MH approach. In the case of noisy data, we have shown that the sequential approach only outperforms standard MH-MCMC when step-sizes are relatively large. The inverse problem defined in Chapter 5 is an example of an inverse problem which yields a posterior which has no obvious optimal step-size. In this situation, the sequential MCMC algorithm can be a practical method for efficiently generating samples from the posterior.

## 6.2 Future Work

The sequential approach described in this work is very general and could be applied to a variety of other problems beyond petroleum reservoirs. In principle, any problem in which the model is time-dependent and the time series data has some redundancy is a candidate for the method. Examining application of the method to other problems and application areas should be the focus of future work.

Additionally, here, we have implemented the sequential approach using a rejection stage at every time-step of the model. It may be more natural in some problems to only perform a rejection stage after multiple time-steps of the model have been computed for a proposed jump. This idea could also be helpful if one is using the sequential approach with a large step-size to generate samples in order to explore parameter space and determine a more appropriate step-size for a standard MH algorithm. We have shown that, for a Gaussian posterior with known optimal step-size, the sequential approach is less efficient than the standard MH approach. In this case however, one could use the sequential approach described here with a rejection stage at every time-step. As the step-size is tuned to the posterior, one could gradually increase the number of model solves evaluated before a rejection stage is performed. We expect that this scheme would allow the use of initial time-steps to inform the algorithm while yielding fewer false rejections as the step-size potentially decreases in

some directions.

Finally, although we do not show the results here, the sequential approach seems to be very efficient for burn-in periods when the chain is initialized far from a mode of the posterior. This is likely because when on starts very far from regions of high posterior probability, a proposed jump away from a region of high probability is likely to yield a degradation in the likelihood function for all time-steps. Conversely, a proposed jump towards a region of high posterior probability is likely to yield an increase in the likelihood function for each time-step so that false rejections are unlikely to occur. This is another area in which future work is required.

# Bibliography


[1] N Schwenk, B Flemisch, R Helmig, and BI Wohlmuth. Dimensionally reduced flow models in fractured porous media. *Comput. Geosci*, 16:277–296, 2012.

[2] Knut-Andreas Lie. *An introduction to reservoir simulation using MATLAB/GNU Octave*. Cambridge University Press, 2019.

[3] Dean S Oliver and Yan Chen. Recent progress on reservoir history matching: a review. *Computational Geosciences*, 15(1):185–221, 2011.

[4] Zhangxin Chen, Guanren Huan, and Yuanle Ma. *Computational methods for multiphase flows in porous media*, volume 2. Siam, 2006.

[5] Vincent Martin, Jérôme Jaffré, and Jean E Roberts. Modeling fractures and barriers as interfaces for flow in porous media. *SIAM Journal on Scientific Computing*, 26(5):1667–1691, 2005.

[6] Philippe Angot, Franck Boyer, and Florence Hubert. Asymptotic and numerical modelling of flows in fractured porous media. *ESAIM: Mathematical Modelling and Numerical Analysis*, 43(2):239–275, 2009.

[7] Elyes Ahmed, Jérôme Jaffré, and Jean E Roberts. A reduced fracture model for two-phase flow with different rock types. *Mathematics and Computers in Simulation*, 137:49–70, 2017.

[8] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. *Automated Solution of Differential Equations by the Finite Element Method*. Springer, 2012.

[9] Alessio Fumagalli and Anna Scotti. A numerical method for two-phase flow in fractured porous media with non-matching grids. *Advances in Water Resources*, 62:454–464, 2013.

[10] Halvor M Nilsen, Josten R Natvig, Knut-Andreas Lie, et al. Accurate modeling of faults by multipoint, mimetic, and mixed methods. *SPE Journal*, 17(02):568–579, 2012.

[11] Christophe Geuzaine and Jean-François Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11):1309–1331, 2009.

[12] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. Chapman and Hall/CRC, 2006.

[13] Andrew M Stuart. Inverse problems: a bayesian perspective. *Acta numerica*, 19:451–559, 2010.

[14] Patrick Billingsley. *Probability and measure*. John Wiley & Sons, 2008.

[15] Skipper Seabold and Josef Perktold. Statsmodels: Econometric and statistical modeling with python. In *Proceedings of the 9th Python in Science Conference*, volume 57, page 61. Scipy, 2010.

[16] Yuttapong Thawornwattana, Daniel Dalquen, Ziheng Yang, et al. Designing simple and efficient markov chain monte carlo proposal kernels. *Bayesian Analysis*, 13(4):1033–1059, 2018.